Position-Based Multicast for Mobile Ad-hoc Networks

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Dipl.-Inf. Matthias Transier
aus Heidelberg

Mannheim, 2008

Dekan:	Prof. Dr. Matthias Krause, Universität Mannheim
Referent:	Prof. Dr.-Ing. Wolfgang Effelsberg, Universität Mannheim
Korreferent:	Prof. Dr.-Ing. Dr.-Ing. E. h. Norbert J. Fliege, Universität Mannheim

Tag der mündlichen Prüfung: 3. Juli 2008

Abstract

In general, routing protocols for mobile ad-hoc networks (MANETs) can be classified into topology-based protocols and position-based protocols. While for unicast routing many proposals for both classes exist, the existing approaches to multicast routing basically implement topology-based algorithms and only a few of them make use of the geographic positions of the network nodes. These have in common that the sending node has to precalculate the multicast tree over which the packets are distributed and store it in each packet header. This involves two main issues: (a) These approaches are not very flexible with regard to topological changes which abandons the advantages that position-based routing has against topology-based routing, and (b) they do not scale with the number of receivers, since every one of them has to be named in the packet header.

This thesis solves these issues and further advances position-based multicast routing. Position-Based Multicast (PBM) enhances the flexibility of position-based multicast routing by following the forwarding principle of position-based unicast routing. It transfers the choice of the next hops in the tree from the sender to the forwarding nodes. Based on the positions of their neighboring nodes, these are able to determine the most suitable next hop(s) at the moment when the packet is being forwarded.

The scalability with respect to the number of receiving nodes in a group is solved by Scalable Position-Based Multicast (SPBM). It includes a membership management fulfilling different tasks at once. First, it administers group memberships in order to provide multicast sources with information on whether nodes are subscribed to a specific group. Second, it implements a location service providing the multicast sources with the positions of the subscribed receiver nodes. And third, it geographically aggregates membership data in order to achieve the desired scalability. The group management features two modes of operation: The proactive variant produces a bounded overhead scaling well with the size of the network. The reac-

tive alternative, in contrast, reaches low worst-case join delays but does not limit the overhead.
Contention-Based Multicast Forwarding (CBMF) addresses the problems that appear in highly mobile networks induced by outdated position information. Instead of basing forwarding decisions on a perception that may no longer be up to date, the packets are addressed only to the final destination; no explicit next hops are specified. The receiving nodes, which are candidate next hops, then decide by means of contention which of them are the most suitable next hop(s) for a packet. Not only is the decision made based on the most currently available data, but this procedure also saves the regular sending of beacon messages, thus reducing the overhead.
The lack of multicast congestion control is another unsolved problem obstructing high-bandwidth data transmission. Sending out more and more packets to a multicast group lets the performance decrease. Backpressure Multicast Congestion Control (BMCC) takes care that the network does not need to handle more packets than it is able to. It achieves this by limiting the packet queues on the intermediate hops. A forwarder may not forward the next packet of a stream before it has noticed—by overhearing the transmission of the next hop—that the previous packet has succeeded. If there is congestion in an area, backpressure is implicitly built up towards the source, which then stops sending out packets until the congestion is released. BMCC takes care that every receiving node will receive packets at the same rate. An alternative mode of operation, BMCC with Backpressure Pruning (BMCC-BP) allows the cutting of congested branches for single packets, permitting a higher rate for uncongested receivers.
Besides presenting protocols for multicast communication in MANETs, this thesis also describes implementations of two of the above-mentioned protocols. The first one is an implementation of SPBM for the Linux kernel that allows IP applications to send data via UDP to a group of receivers in an ad-hoc network. The implementation resides between the MAC layer and the network/IP layer of the network stack. It is compatible with unmodified standard kernels of versions 2.4 and 2.6, and may be compiled for x86 or ARM processor architectures. The second implementation is an implementation of CBMF for the ScatterWeb MSB430 sensor nodes. Due to their low-level programmability they allow an integration of the routing protocol with the medium access control. The absence of periodic beacon messages makes the protocol especially suitable for energy-constrained sensor networks. Furthermore, other constraints like limited memory and computational power demand special consideration as well.

Zusammenfassung

Im Allgemeinen können Routingprotokolle für mobile Ad-hoc-Netzwerke (MANETs) in topologie- und positionsbasierte Protokolle eingeteilt werden. Während es im Bereich des Unicastroutings zahlreiche Vertreter beider Klassen gibt, beschränken sich die existierenden Multicastprotokolle im Wesentlichen auf topologiebasierte Ansätze, und nur einige wenige nutzen Positionsinformationen. Diese Verfahren haben gemeinsam, dass der sendende Netzwerkknoten den Multicastbaum, der für die Paketzustellung genutzt werden soll, vorberechnet und ihn in jedem versendeten Paket ablegt. Dieses Vorgehen bringt zwei Probleme mit sich: (a) Die Verfahren reagieren nicht flexibel auf topologische Veränderungen und verschenken damit die Vorteile des positionsbasierten Routings. (b) Sie skalieren nicht gut mit der Anzahl der Empfänger, da jeder einzelne von ihnen im Paketkopf aufgeführt werden muss.

Diese Dissertation löst die angesprochenen Probleme und bringt das positionsbasierte Multicastrouting weiter voran. Position-Based Multicast (PBM) erhöht die Flexibilität des positionsbasierten Multicastrouting, indem es das Prinzip der Weiterleitung wie es für Unicastrouting angewandt wird für Multicast adaptiert. Es verlagert die Auswahl der Routen im Baum vom Sender zu den weiterleitenden Knoten im Netzwerk. Basierend auf den Positionen der benachbarten Knoten können diese die zum Zeitpunkt der Weiterleitung am besten geeigneten Nachbarn bestimmen.

Die Skalierbarkeit in Bezug auf die Anzahl der Empfänger in der Gruppe wird durch Scalable Position-Based Multicast (SPBM) erreicht. Dieses Protokoll schließt eine Mitgliedschaftsverwaltung ein, die mehrere Aufgaben erfüllt. Erstens verwaltet sie die Mitgliedschaften und versorgt die Quellknoten mit Informationen darüber, ob eine bestimmte Gruppe Mitglieder enthält. Zweitens implementiert sie einen Ortungsdienst, der die Quellen mit den Positionen der angemeldeten Empfänger versorgt. Und drittens aggregiert sie die Mitgliedschaftsinformationen geographisch, um die

gewünschte Skalierbarkeit zu ermöglichen. Die Gruppenverwaltung unterstützt zwei verschiedene Modi: Die proaktive Variante produziert einen beschränkten Overhead, der gut mit der Netzwerkgröße skaliert. Im Gegensatz dazu erreicht die reaktive Alternative auch im schlechtesten Fall niedrige Verzögerungszeiten für einen Gruppenbeitritt, garantiert dabei allerdings nicht die Begrenzung des Overheads.

Contention-Based Multicast Forwarding (CBMF) geht die Probleme an, die in hochmobilen Netzwerken durch veraltete Positionsinformationen entstehen. Statt die Weiterleitungsentscheidung auf Basis einer Wahrnehmung zu fällen, werden die Pakete nur an die Zielpositionen adressiert; es werden keine expliziten Weiterleiter angegeben. Die Kandidaten für die Weiterleitung entscheiden selbst in einem Wettbewerb, wer von ihnen am besten geeignet ist. Die Entscheidung wird dadurch nicht nur auf Basis der bestmöglichen Daten getroffen, dieses Vorgehen spart zusätzlich das regelmäßige Versenden von Positionsnachrichten ein und reduziert so den Overhead.

Ein weiteres ungelöstes Problem, das der Datenübertragung mit hoher Bandbreite entgegensteht, ist das Fehlen einer Staukontrolle für Multicast in Ad-hoc-Netzwerken. Das Senden von immer mehr Paketen an eine Multicastgruppe verringert den Durchsatz. Backpressure Multicast Congestion Control (BMCC) sorgt dafür, dass das Netzwerk nicht mehr Pakete aufnehmen muss als es kann. Das wird erreicht, indem die Paketwarteschlangen auf den Knoten unterwegs streng limitiert werden. Ein Knoten darf kein weiteres Paket eines Datenstroms weiterleiten, bevor er nicht — durch das Mithören der Übertragung des nächsten Knotens — mitbekommen hat, dass das vorherige Paket bereits auf dem Weg zum übernächsten Knoten ist. Dadurch wird, wenn ein Gebiet überlastet ist, implizit ein Rückdruck in Richtung der Quelle aufgebaut, die daraufhin das Versenden weiterer Pakete aussetzt. BMCC sorgt dafür, dass jeder Empfänger Pakete mit der gleichen Rate erhält. Ein alternativer Modus, BMCC mit Backpressure Pruning (BMCC-BP), erlaubt das Abschneiden überlasteter Zweige für einzelne Pakete, wodurch Empfänger, die nicht überlastet sind, die Möglichkeit bekommen, Daten mit einer höheren Datenrate zu empfangen.

Neben der Vorstellung von Protokollen für die Multicastkommunikation in MANETs beschreibt diese Dissertation auch die Implementierungen zweier oben erwähnter Protokolle. Die erste ist eine Implementierung von SPBM für den Linuxkern, die es IP-Anwendungen erlaubt, Daten per UDP an eine Gruppe von Empfängern in einem Ad-hoc-Netzwerk zu senden. Die Implementierung ist im Schichtenmodell zwischen der

Medienzugriffs- und der Netzwerkschicht angesiedelt. Sie ist mit den unmodifizierten Standardkernen der Versionen 2.4 und 2.6 kompatibel und kann sowohl für x86- also auch für ARM-Prozessorarchitekturen kompiliert werden. Die zweite Implementierung ist eine Implementierung von CBMF für die Sensorknoten ScatterWeb MSB430. Da sie sich auf unterster Netzwerkebene programmieren lassen, erlauben sie eine Integration des Routingprotokolls mit der Medienzugriffskontrolle. Das Fehlen periodischer Positionsnachrichten erleichtert den Einsatz des Protokolls in Sensornetzwerken mit beschränkten Energievorräten. Darüberhinaus erfordern auch andere Einschränkungen wie ein begrenzter Speicherplatz und eine eingeschränkte Rechenleistung besondere Beachtung.

Danksagungen

Diese Arbeit wäre nicht möglich gewesen ohne die Unterstützung vieler. Dafür möchte ich im Einzelnen danken:

Zu allererst Herrn Prof. Dr. Wolfgang Effelsberg, der meine Arbeit betreut hat und Prof. Dr. Martin Mauve, der das Projekt auf den Weg brachte, aus dem diese Dissertation entstand.

Ebenso der Prüfungskommission, Prof. Dr. Dr. Norbert Fliege für die freundliche Übernahme des Korreferats, dem Prüfungsvorsitzenden Prof. Dr. Felix Freiling sowie Prof. Dr. Heiner Stuckenschmidt, der sich als dritter Prüfer zur Verfügung gestellt hat.

Meinen Kollegen am Lehrstuhl, allen voran Holger Füßler, dem ich sowohl meine Arbeit betreffend als auch persönlich viel zu verdanken habe. Weiterhin Marcel Busse, Dirk Farin, Benjamin Guthier, Thomas Haenselmann, Thomas King, Stephan Kopf, Fleming Lampi, Hendrik Lemelson, Christian Liebig, Michael Möske, Nicolai Scheele, Robert Schiele, Ralf Schmitz, Sascha Schnaufer, Moritz Steiner, Tonio Triebel, Jürgen Vogel und Jörg Widmer, die meine Zeit am Lehrstuhl zu dem Erlebnis gemacht haben, das sie war.

Für die gute Zusammenarbeit im DFG Schwerpunktprogramm und darüberhinaus danke ich Christian Lochert, Wolfgang Kieß und Björn Scheuermann aus Düsseldorf sowie Daniel Görgen aus Trier. Weiterhin meinen anderen Co-Autoren Marc Torrent-Moreno und Prof. Dr. Hannes Hartenstein aus Karlsruhe, Prof. Dr. Peter Sturm aus Trier, Andreas Festag von NEC, Roland Krüger und Angelika Leibscher.

Den Sekretärinnen des Lehrstuhls Betty Haire-Weyerer und Ursula Eckle, sowie Sabine Baumann als ihre Vertretung, und unserem Sysadmin Walter Müller. Besonderer Dank gilt dabei Betty Haire-Weyerer, die neben unseren zahlreichen englischen Veröffentlichungen auch diese Arbeit Korrektur gelesen und mir dabei viele wertvolle Hinweise gegeben hat.

DANKSAGUNGEN

Meinen Diplomanden, insbesondere Thomas Butter und Jan Kästle, den Studien- und Bachelorarbeitern, sowie den Hilfskräften, insbesondere Hans Pfründer und — für die Unterstützung bei der Verwaltung der Lehrstuhl- und Instituts-Webseiten — Marco Szabo und Nadine Benedum.
Sina Borgsen, Martina Corsten und Christine Kaufmann, die mir geholfen haben, die Wartezeit auf die Begutachtung zu verkürzen.
Und zu guter letzt meinen Eltern, meinem Bruder und meiner Oma für ihre Unterstützung.
Vielen Dank!

Contents

List of Tables

List of Figures

Abbreviations

ABAM	Associativity-Based Ad hoc Multicast
ABR	Associativity-Based Routing
ACK	Acknowledgment
ACLK	Auxiliary Clock
ADC	Analog-to-Digital Converter
ADMR	Adaptive Demand-Driven Multicast Routing
AMRIS	Ad hoc Multicast Routing Protocol Utilizing Increasing ID-numbers
AMRoute	Ad hoc Multicast Routing Protocol
AODV	Ad hoc On-Demand Distance Vector Routing
AODV-UCSB	AODV implementation of the University of California at Santa Barbara
AODV-UU	AODV implementation of Uppsala University
ARM	Advanced RISC Machines Ltd.
ASK	Amplitude-Shift Keying
BEM	Bandwidth-Efficient Multicast
BLR	Beacon-Less Routing Algorithm
BMCC	Backpressure Multicast Congestion Control
BMCC-BP	BMCC with Backpressure Pruning

BoF	Birds-of-a-Feather
CAMP	Core-Assisted Mesh Protocol
CBDV	Contention-Based Distance Vector Routing
CBF	Contention-Based Forwarding
CBM	Content-Based Multicast
CBMF	Contention-Based Multicast Forwarding
CCR	Capture Compare Register
CDF	Cumulative Distribution Function
CEDAR	Core-Extraction Distributed Ad hoc Routing
CGM	Clustered Group Multicast
CoRe	Communications Research group at Uppsala University
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA	Carrier Sensing Multiple Access
CTF	Clear to Forward
CTS	Clear to Send
CXCC	Cooperative Cross-Layer Congestion Control
DCMP	Dynamic Core-Based Multicast Routing Protocol
DDM	Differential Destination Multicast
DNS	Domain Name System
DREAM	Distance Routing Effect Algorithm for Mobility
DSM	Dynamic Source Multicast
DSR	Dynamic Source Routing
DVMRP	Distance Vector Multicast Routing Protocol

EEPROM	Electrically Erasable Programmable Read-Only Memory
ERS	Expanding Ring Search
FGMP	Forwarding Group Multicast Protocol
FGMP-RA	FGMP with Receiver Advertising
FGMP-SA	FGMP with Sender Advertising
GLS	Grid Location Service
GNU	GNU is not Unix
GOAFR	Greedy Other Adaptive Face Routing
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GRSS	Geographical Region Summary Service
GUI	Graphical User Interface
HP	Hewlett-Packard Company
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMAHN	Integrated Multicast for Ad-hoc Networks
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISM	Industrial, Scientific, and Medical Frequency Band
KAL	Keep Alive
LAM	Lightweight Adaptive Multicast
LAN	Local Area Network
LAR	Location-Aided Routing
LBM	Location-Based Multicast

LGT	Location-Guided Tree
MAC	Medium Access Control
MANET	Mobile Ad hoc Network
MAODV	Multicast Ad hoc On-Demand Distance Vector Routing
MBONE	Multicast Internet Backbone
MCEDAR	Multicast Core-Extraction Distributed Ad hoc Routing
mDNS	Multicast Domain Name System
MMAC	Multicast Medium Access Control
MMNET	Mobile Mesh Network
MOSPF	Multicast Open Shortest Path First
MRD	Modified Random Direction Mobility Model
MZR	Multicast Routing Protocol Based on Zone Routing
NACK	Negative Acknowledgment
NAV	Network Allocation Vector
NMEA	National Marine Electronics Association
NSMP	Neighbor Supporting Multicast Protocol
ODMRP	On-Demand Multicast Routing Protocol
OLAM	On-Demand Location Aware Multicast
OLSR	Optimized Link State Routing
OOK	On-Off Keying
PBM	Position-Based Multicast
PBR	Position-Based Routing
PDR	Packet Delivery Ratio
PIM	Protocol Independent Multicast

PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
RAM	Random Access Memory
RBM	Reservation-Based Multicast
RFA	Request For Acknowledgment
RFC	Request For Comments
RISC	Reduced Instruction Set Code
RLS	Reactive Location Service
RP	Rendezvous Point
RREP	Route Reply
RREQ	Route Request
RTF	Request To Forward
RTS	Request To Send
RWP	Random Waypoint Mobility Model
SMCLK	Sub-System Master Clock
SPBM	Scalable Position-Based Multicast
SPBM-BC	SPBM using Broadcast
SRMP	Source-Routing-Based Multicast Protocol
ST-WIM	Shared Tree Wireless Multicast
TCP	Transmission Control Protocol
TORA	Temporally-Ordered Routing Algorithm
UDP	User Datagram Protocol
UNIRP	Unicast Routing Protocol
WLAN	Wireless Local Area Network

CHAPTER 1

Introduction

In today's world everybody and everything are connected. Everyone is using email and surfing the Web on a daily basis. From young to great ages, life without the Internet is just not conceivable anymore. Apart from some radio or satellite links, conventionally, the Internet and networks in general have for the most part been wired. Beginning with the commercial launch of mobile terminal devices of various kinds, more and more people are connected on the go. Up to now, such mobile devices have been based on a supporting infrastructure like cellular networks or wireless access points. During recent years, ad-hoc communications, i. e., communications without any fixed infrastructure, have attracted considerable research interest. Mobile Ad-hoc Networks (MANETs) pose completely new challenges with regard to algorithms for communication within such networks as opposed to conventional wired networks. On one hand, wireless communication is much more prone to transmission errors than is wired communication, and, on the other hand, mobility introduces a whole new dimension of dynamics. Both issues have to be thoroughly resolved during the design of new protocols for MANETs. But, let us start at the beginning and briefly outline the history of MANETs . . .

1.1 Mobile Ad-hoc Networks

1.1.1 Packet Radio Networks

The roots of mobile wireless ad-hoc networks can be found in the packet radio networks that evolved in the 1970's. Before that time, only point-to-point links had been used for wireless data transmission. These links were based on the AX.25 data link layer on amateur radio frequencies. Its current version is described in [19]. The transmission speed achieved by these links is only a few kilobits per second. Their purpose is to support text transmission, like electronic mail or articles in bulletin boards.

The first step towards ad-hoc networks was the extension of point-to-point links to a broadcast shared medium. The basis for the development was the ALOHA protocol described by Abramson in 1970 [15, 16]. This simple protocol allows all participants in the network to send their data packets just when they want to. In the event of a collision, the affected sender will wait a random backoff time and then try again. This protocol assumes that every participant is in radio range to one another and is thus able to communicate with and listen to the communications of all other participants. ALOHA achieves a maximum relative throughput of around 18 %, i. e., during 18 % of the time, successful transmissions take place, while the remaining 82 % is wasted for idle times, collisions, and retransmissions. In 1972, Roberts extended the ALOHA protocol with the concept of *time slots* [158, 159]. It requires a synchronized time on all terminals and defines points of time when a transmission may start. Thus, collisions may only occur at beginnings of time slots. Packets that have successfully started to be transmitted are sure to be completely transmitted without collisions. This concept increases the maximum relative throughput by the factor of two. Since then, it has been used in nearly all medium access protocols for the wireless medium. In the same work, Roberts investigated another important fact, the *packet capture*. This effect allows a receiver to successfully receive a collided packet if its signal strength is sufficiently greater than that of the other colliding signals.

The next step towards the ad-hoc networks of today was the introduction of *carrier sense multiple access* (CSMA) by Kleinrock in 1975 [111]. While terminals with ALOHA just start to send when they are ready, with CSMA they sense the medium before transmitting a packet. This reduces the possibility for collisions and further increases the maximum possible throughput. If the medium is busy when a terminal wants to start a transmissions, two different protocols are possible: *nonpersistent* and *p-persistent* CSMA. With the first one, the transmission attempt is repeated after a given time. In p-persistent CSMA, however, with a probability of p the transmission is started immediately after the medium has become idle again.

All the protocols described so far have in common that all terminals have to be within communication range of each other. In 1977, Kahn extended the ALOHA protocol to encompass *multi-hop* communication [103]. His system contains fixed stations to organize the network, fixed repeaters to connect terminals that are not able to communicate directly, and mobile terminals to be carried around by users. Among other things, Kahn addresses authentication issues and the possible coexistence of other systems on the same frequency band. In [104], Kahn et al. provide an extensive and detailed overview of packet radio technology at that point in time, including *spread spectrum* transmission, which is also an important part of the MANET technology used today. It assures that, e. g., wireless LAN can be used on an ISM frequency band which is also open to any other application.

A few years later, Takagi and Kleinrock [173] investigated multi-hop packet radio networks with randomly distributed terminals. They discovered that the dominance of CSMA over ALOHA is much smaller in such networks than in one-hop networks. The identified reason is so-called *hidden terminals* [178]. The hidden-terminal effect can cause collisions at a receiver that are not recognized by the involved senders if these are not within range of each other. The paper can be seen as a milestone towards position-based routing for MANETs (see Chapter 2.1.1). It already contains theoretical analyses of a greedy position-based routing strategy.

In 1987, Jubin and Tornow [102] stated that at that time existing protocols were able to handle mobile networks with up to 50 terminals. These protocols include mechanisms for dynamic routing, congestion control, and channel allocation under varying conditions. A main aspect of packet radio networks is still the connection of mobile nodes to the Internet. In the same issue of the journal, Shacham and Westcott [166] discussed future directions for packet radio networks: hierarchical routing for large networks, the usage of multiple channels for a better handling of high network traffic, and the adaptation of transmission and protocol parameters in order to cope with changing network conditions.

1.1.2 Current Technology

The boom in mobile ad-hoc networks research during the last few years started in 1996, when the Internet Engineering Task Force (IETF) held a so-called Birds-of-a-Feather (BoF) meeting in Montreal. Its purpose was the formation of a new working group with the title "Mobile Mesh Networks (MMNET)". One year later, the new IETF working group "Mobile Ad hoc Networks (MANET)" was officially announced, and the first working group meeting took place in August 1997 in Munich [131]. So far, the working group has published one request for comments (RFC) regarding routing protocols for MANETs in general [53] and four RFCs about unicast routing protocols in MANETs [151, 49, 145, 100]. For multicast routing, there is currently only one draft entitled "Simple Multicast Forwarding for MANET" [130]. There have been a number of drafts of multicast protocols [123, 96, 195, 98, 162, 94], but they have since become outdated, and none of them has ever reached the status of an RFC.

The majority of ad-hoc networks today is based on the IEEE 802.11 suite [87, 88, 89] of data link and physical layers, also known as *wireless LAN* (WLAN). The Wi-Fi alliance [13] is an organization consisting of over 300 manufacturers of WLAN hardware. Their goal is to supervise the adherence of IEEE 802.11 hardware to standards and thus assure interoperability. While nowadays nearly every new notebook computer has a built-in WLAN interface, these are used mainly for connectivity to Internet access points. Mobile ad-hoc networks are still not widely used.

There is, however, a number of static ad-hoc networks, also known as *mesh networks* [90], that have recently been deployed. They are mainly used for experimentation purposes with the goal to provide users in metropolitan areas with wireless Internet access [28, 9]. But there are already a few commercial suppliers of mesh networks (e. g., [12, 4]).

1.2 Multicast

This thesis explores multicast in mobile ad-hoc networks. Multicast is a form of group communication used when a data source disseminates the same information to a group of receivers. Examples include video conferencing, TV over the Internet, collaboration applications like shared whiteboards, Internet chats, and others. Instead of transmitting the same data to each of the receivers separately, multicast follows the approach of dissemination trees: Each data packet is sent out only once, and is afterwards duplicated in the network on its way to the destinations. This procedure avoids duplicate transmissions of the same data over the same link and thus saves bandwidth in the network, especially at the data sources.

The first research papers on multicast routing were published in the early 1980's. In his PhD thesis [189], Wall discusses the adaptation of existing algorithms for broadcasting to all nodes in a network to the problem of *selective broadcast*—the communication with a selected set of destinations. At this time, broadcast algorithms were still a hot topic in network research [35]. The term *multicast* first appeared in 1984, when Aguilar [18] described how multidestination routing over shortest paths could be incorporated into the Internet Protocol (IP). In 1985, Cheriton and Deering [44, 57] presented a protocol to be used for multicast routing in the Internet. In the following years, Deering continued his work on this topic [58, 60] and published his PhD thesis [59].

The described research established a basis for the implementation of the Multicast Internet Backbone (MBONE) [63]. In 1992, this backbone of multicast enabled routers was set up. Mainly universities were connected for the purpose of experimentations with multicast. The employed protocols were the Distance Vector Multicasting Routing Protocol (DVMRP) [188], the Multicast Open Shortest Path First (MOSPF) protocol [141] and, later, the Protocol Independent Multicast (PIM) [64] with variants for dense networks (PIM-DM) [17] and sparse networks (PIM-SM) [66]. PIM uses routing information that has been collected by any associated unicast routing protocol. In today's Internet, PIM-SM is the protocol that is mostly used. [29] gives a short overview of the current state of research in the area of Internet multicast.

In local area networks (LAN), multicast is used for a variety of application protocols. Especially in the field of network autoconfiguration, multicast plays an important role. Apple Bonjour [1], the multicast domain name system (multicast DNS or mDNS) [45], and zero configuration networking [46] are based on multicast and allow the quick and easy setup of small networks without any manual configuration.

As mentioned before, multicast routing did not receive the attention in the IETF MANET group that unicast routing did. Chapter 2 will give an overview and a classification of existing multicast routing protocols for MANETs.

1.3 Overview of this Thesis

Chapter 2 describes the state of the art in research on multicast routing protocols for MANETs. A classification and short descriptions of the most important protocols give an introductory overview. The following Chapter 3 describes a first straight-forward proposal for position-based multicast that is derived from position-based unicast forwarding. It shows the shortcomings of such an approach and serves as a basis for the development of the following protocols. Chapter 4 presents Scalable Position-Based Multicast (SPBM), comprising a group management mechanism and a multicast forwarding strategy. While this approach scales well for networks with large numbers of nodes, it has problems with mobility. These are overcome by Contention-Based Multicast Forwarding (CBMF), presented in Chapter 5.

Following the different routing protocols, Chapter 6 introduces a congestion control mechanism based on SPBM, improving the performance of SPBM in heavy load situations considerably.
While the protocol implementations up to this point are altogether implementations for network simulations, Chapter 7 describes two real proof-of-concept implementations of SPBM—one for the Linux kernel, to be used for multicast forwarding in real-world MANETs, and a second one for FU Berlin sensor nodes based on an MSP430 processor.
Finally, Chapter 8 concludes the thesis, summarizing the most important results and contributions.

CHAPTER 2

State of the Art in Ad-hoc Multicast

There are quite a number of multicast protocols for ad-hoc networks. Beginning in 1995 [50], many different approaches have been proposed. This chapter will start by describing some characteristics that allow a classification of the proposals. In the following, the algorithms of existing protocols are outlined and classified into the previously defined classes. Finally, this chapter concludes by summing up the open questions regarding position-based multicast for mobile ad-hoc networks.

2.1 Classification Characteristics

2.1.1 Topology- vs. Position-based Protocols

The classification of a routing protocol according to whether it is topology-based or position-based depends on how the protocol calculates its routes. Routing decisions in ad-hoc networks are always based on neighbor relations in the network. These relations can be used in two different ways.

Topology-based Protocols

Topology-based protocols use the information about existing links between neighboring nodes to perform packet forwarding. This information can be seen as a topology graph of the network, and a routing algorithm in this context is a type of distributed graph algorithm that searches a path from a source to a destination node. Principally, a route has to be discovered prior to starting the actual packet transmission. Local topology information alone does not give any hints on which of the neighboring nodes a packet should be sent to—unless one of the neighboring nodes is the destination itself. This introduces a potential delay for packets that should be sent before any routes have been discovered. Furthermore, if a link within a previously discovered route breaks due to topological changes caused by node movement or radio fluctuations, the corresponding route has to be repaired, or a new route has to be found.

Position-based Protocols

Position-based protocols make use of physical node positions to overcome the drawbacks of topology-based approaches. These protocols require the participating nodes to be aware of their geographic positions. This can be accomplished by means of GPS [105] or some GPS-free technique [41], especially for indoor scenarios [84, 109]. Position-based protocols do not rely on discovering routes that are used for all subsequent packets, instead, they perform forwarding on a hop-by-hop basis. Packets are routed by means of the current node's position, the neighbors' positions and the destination's position. This position information is sufficient to decide which next hop should receive a packet. In *Greedy Forwarding*, one form of position-based routing, nodes periodically exchange beacon messages containing their own position. In this way, every node is able to build a table containing information on all its neighboring nodes within radio range. If a node wants to send a packet, it compares the distances of all its neighbors to the desired destination and picks the neighbor closest to the destination. Sending the packet to this neighbor will yield the maximum progress towards the destination. Hop by hop, the packet will be forwarded and eventually reach its final destination.

While a node's own position can be obtained by positioning services and the neighbors' position by means of beaconing, the position of the destination can be provided by a *Location Service*. There are a number of algorithms for location services. A straight-forward one is the Reactive Location Service (RLS) [117]. It works as follows: The querying node creates a request packet including its own position as the source address and the ID of the destination. It then broadcasts the packet to all its neighbors in radio range. Each of them again broadcasts the packet. To avoid that the packet circulates in the network forever, a node receiving the same packet for the second time simply discards it. The re-broadcasting of a received packet is delayed for a random backoff time in order to sidestep the broadcast storm problem [142] that is caused by collisions of packets simultaneously rebroadcast by different forwarders. Eventually, the destination node receives the request packet and creates a response packet. This response contains the inquired position as source address and the ID and position of the querying node, which were contained in the query packet, as destination. This response packet is then sent back to the querying node using position-based unicast as described above.

One problem that can occur with position-based routing is the existence of voids, i. e., regions where no node is located, so that packets reach a local optimum on their way to the destination. Figure 2.1 shows such a case. A packet that is on its way from node S to node D gets stuck when it reaches node A. There is no node that is located within radio range r to A and at the same time located closer to D. In order to get out of such a local optimum, the packet has to accept the loss of some progress before it can proceed again. For these cases, a variety of *Recovery Strategies* exists that are employed whenever a greedy routing algorithms gets stuck [106, 36, 125, 68, 110].

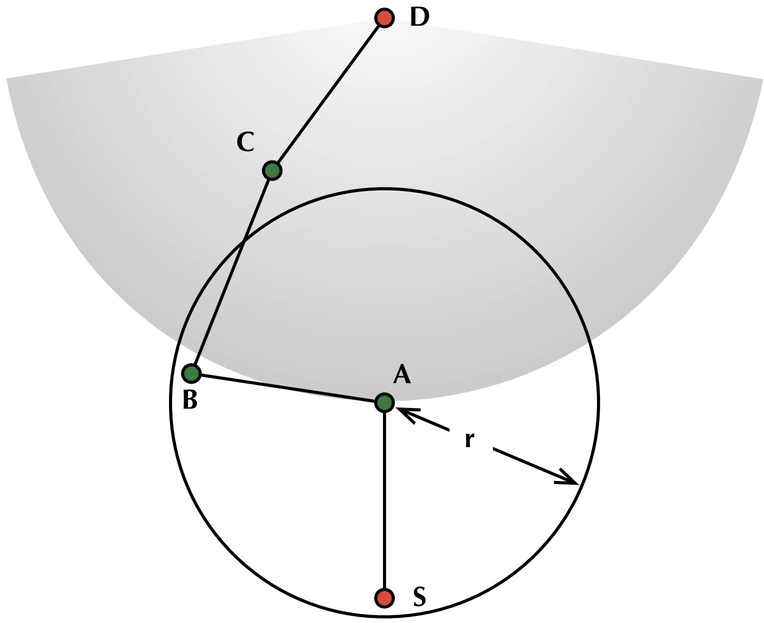


Figure 2.1: *Local optimum in greedy forwarding*

A Word on Unicast Routing

Unicast routing protocols for mobile ad-hoc networks are generally classified into topology-based and position-based protocols. [163] gives a survey of the most important topology-based representatives, while [135, 172, 76] describe position-based approaches. See also [67], which gives a comprehensive overview of geographical routing for a specific application scenario, namely vehicular ad-hoc networks.

The vast majority of existing multicast protocols for ad-hoc networks are topology-based. They are described in Section 2.2.1, while Section 2.2.2 deals with the protocols based on geographic forwarding.

2.1.2 Tree vs. Mesh Structure

Tree-based Protocols

Tree-based multicast protocols are characterized by the property that they construct a tree for the distribution of data packets. Each node that is part of such a tree has its well-defined successors—unless it is a leaf node. Should part of a tree be affected by topological changes within the network, a tree reconstruction is required.

Tree reconstructions may be performed globally or locally. While a global reconstruction means that the currently used tree is discarded and a new one is built, a local reconstruction is more sophisticated. It defines additional protocol elements that allow reconnection of a tree at the point where it broke. The benefit of this behavior is that the communication overhead induced by the repair procedure can be kept to a minimum.

Delivery trees can be classified into two categories: *source-specific trees* and *shared trees*. A source-specific tree is a tree that is rooted at a specific source and that is used only to distribute packets generated by this source. All other sources have to build and maintain their own trees for their packets. A special kind of source-specific trees are those based on a *rendezvous point*. In these trees it is not the source that is the root of the tree but a rendezvous point (RP). The RP maintains the tree paths to all receivers of its group and handles all packets that are sent to the group. A source that wants to deliver a multicast packet sends its packet to the closest rendezvous point, which then feeds the packet into the delivery tree.

The second category, shared tree-based protocols, uses a single tree per multicast group. Each node is root for its own packets and at the same time tree node for the packets of the other nodes. This reduces management overhead for the case that at least some of the nodes simultaneously act as senders and receivers in the group.

The first proposals of multicast routing protocols for mobile ad-hoc networks were tree-based approaches based on existing multicast protocols for wired networks like PIM-SM [61] (current RFC: [65]). In 1995, [50] described Reservation-Based Multicast, the first multicast protocol for mobile networks.

Mesh-based Protocols

Mesh-based protocols follow a different approach. A mesh of forwarding routers is used to disseminate the data packets. Every node that is part of the mesh can have multiple incoming and outgoing links, i. e., entries in a forwarding table. Whenever a packet arrives at one of the in-going links, it is forwarded to every outgoing link.
On the one hand, a mesh is more robust to topology changes than a tree because it can provide more than one path from a source to each of its destinations. On the other hand, meshes imply an increased overhead for this very reason.

2.1.3 Centralized vs. Decentralized Organization

Regardless of whether designing a tree-based protocol or a mesh-based protocol, the organization of the group management and the data distribution may be done in a centralized or decentralized fashion.

Centralized Organization

Centralized organization means that there is at least one node that acts as a so-called core. It is responsible for maintaining the group memberships or for taking care of the data dissemination. In tree-based protocols, e. g., the rendezvous point is a core for the data dissemination, and every data packet is routed to this rendezvous point before it is sent to the tree of recipients.

Decentralized Organization

Decentralized approaches try to coordinate the nodes in a distributed manner. This means that senders and receivers contact each other directly in order to build up the required data distribution structure. Which of the two, sender or receiver, actually initiates the contact is determined by the specific protocol.

2.1.4 Sender vs. Receiver Initiation

As mentioned, there are two alternative ways in which a protocol can handle the initiation of a multicast session.

Sender Initiation

A *sender initiated* session starts with a node that is willing to send data packets to a multicast group. It announces the start of a new session to the nodes in the network and invites them to join the group. After all nodes have subscribed, the source is able to use the session and send packets to the group.

Receiver Initiation

The second alternative is *receiver initiated* sessions. In this case, receivers announce their subscription to certain multicast groups to all nodes in the network. Accordingly, data sources are able to start sending packets immediately once these are generated by the application, since group memberships are maintained independently of existing sources, and receiver nodes have already subscribed to the group.

2.1.5 Sender- vs. Group-specific Addressing

The addressing of multicast groups may vary depending on how group addresses are specified. This influences the way receivers join a multicast group.

Sender-specific Addressing

In *sender-specific* multicast, receiver nodes subscribe to packets generated by a specific sender. This enables nodes to selectively receive multicast transmissions from specific senders. In scenarios where senders provide content and do not require any feedback from the receivers, this addressing scheme is equivalent to the assignment of group addresses to each single sender.

Group-specific Addressing

The second kind of multicast addressing is the use of *group-specific* addresses. In this scheme, receiver nodes subscribe to a certain multicast address and receive all packets that are sent to this group by any source. Group-specific addressing is the standard for IP multicast in the Internet using PIM [66].

Combined Addressing

The multicast address can also be a tuple of a sender ID and a group address, so that a sender can operate different groups of receivers at the same time. In this case, the receivers join a multicast session by specifying a sender ID *and* a group address. The group address may be used by different senders. An exemplary application scenario for this addressing scheme is a group of nodes that are interested in the same topic, and senders in this group provide messages of different granularity. The receivers may then decide which kind of messages they receive. This hybrid system also allows the reuse of group addresses for different multicast sessions. Different senders may use the same group address while their sessions can still be distinguished by their sender ID.

2.1.6 Hard State vs. Soft State

In most cases network protocols need some kind of state information in order to perform their actions. Principally, there are two different approaches to the management of state.

Hard State

The first one is *hard state*. Using the hard-state approach, a protocol changes its state information only upon a concrete request. For multicast group memberships this could, e. g., mean that a node has to explicitly request the leave from a multicast session by sending a message to the group or sender.

Soft State

The second approach is *soft state*. In this case, state information is volatile. It times out after a period of time unless it is periodically refreshed. For the example of multicast group memberships this means that multicast receivers have to renew their subscription to a group on a regular basis. If they fail to do so, they will be excluded from receiving multicast messages from this group.

There are advantages and drawbacks to both methods. On the one hand, a protocol that is based on hard state reduces the number of messages that have to be sent in static scenarios—if nothing changes, no messages have to be sent. The soft-state approach, on the other hand, is able to handle unstable links: If a node loses the connection to a multicast group, this group will after a period of time stop trying to reach the node when the periodical refresh messages from the node fail to appear—unlike the hard-state protocol, which will keep trying to deliver messages to the lost node.

Some protocols try to combine the advantages of both methods. They, e. g., use hard state for seldom changing settings (like the core of a certain multicast group) and soft state for regularly changing settings (like group memberships).

2.1.7 Proactive vs. Reactive Maintenance

Every protocol has to maintain its state in order to be able to successfully route packets to the destinations.

Proactive Maintenance

The difference between *proactive* and *reactive* protocols is that the former continuously maintain up-to-date state information like, e. g., routing tables. In this way, these protocols are able to immediately start a requested operation. This comes at the cost of a permanent network load induced by proactively sent messages.

Reactive Maintenance

Reactive protocols try to minimize this load by only acting if there is a request and "sleeping" otherwise. The drawback to this approach is that there is a delay before the reactive protocol is able to fulfill a request—it first has to collect the required information. Considering routing protocols, this characteristic means that a proactive protocol always maintains up-to-date routing tables to every possible destination, while a reactive protocol only builds up a route to a destination if there is a packet to be sent to this destination.

Many protocols for ad-hoc networks combine proactive and reactive elements. An optimal trade-off between a low network load and a high responsiveness to requests could be described as follows: A minimal number of proactive messages is used to maintain a basic state that in turn helps to accelerate reactive operations and thus reduces the reaction delay. An example of this is position-based forwarding algorithms as described in 2.1.1: Proactive periodic beacon messages between neighboring nodes are used to collect tables containing the neighbors of a node. If there is a packet to be sent, the node may decide the optimal route based on the previously collected information. Up to this point, the protocol can be seen as a proactive one. But in the case that the packet reaches a local optimum and cannot be forwarded using the proactively collected information, a reactive recovery mechanism can be used to escape this situation (see Section 2.1.1 on position-based protocols).

2.1.8 Use of Flooding

An important property of protocols for ad-hoc networks in general is the way flooding is used. In order to reach all nodes in an ad-hoc network, i. e., to broadcast a message to all nodes in the network, some kind of flooding is needed. Flooding is an expensive task in ad-hoc networks since it has to be done in a distributed manner and is thus hard to optimize. The simplest approach is that every node repeats the broadcast message once. This linearly scales with the number of nodes in the network and induces the *broadcast storm problem* [142]. Various proposals have been made to alleviate this problem [190] but flooding remains an expensive task, especially if a protocol requires data packets to be broadcast to the entire network.
Regarding multicast group management and routing protocols for ad-hoc networks, three characteristics of flooding can be distinguished:

Type of flooded packets: As mentioned, packets to be flooded can either be control packets or data packets. While control packets usually have sizes of some tens of bytes, data packets may contain up to over two kilobytes of data when using IEEE 802.11 [87]. Thus, periodic flooding of data packets should generally be avoided.

Occasion of flooding: The reasons for a packet to be flooded may be different. Some protocols try to minimize the amount of flooded messages by only requiring messages to be broadcast that build up or reconstruct a distribution structure. Those messages appear on an irregular basis and only if needed. Other protocols make use of periodically flooded messages—either control or data packets—stressing the entire network with a continuous load.

Type of flooding: The way flooding is performed also affects the load induced in the network. In contrast to a *broadcast* which is intended to reach all nodes in the entire network, the following kinds of flooding are restricted to only a selection of nodes.

Scoped flooding limits the number of hops a message is allowed to traverse during the flooding process. This topologically limits the region where the message is delivered. Very similar is *geographically scoped broadcast*, which does not specify a maximum hop count but a geographical region within which the message shall be broadcast. A special and well-known form of scoped flooding is the *expanding ring search* (ERS). When looking for a certain node, ERS uses a series of scoped flooding attempts with growing hop limits for each attempt until the desired node is reached. The idea is to prevent the flooding of unnecessarily large regions when a smaller region is sufficient. The trade-offs of ERS have been analyzed in [80].

Restricted flooding is limited to a defined group of nodes which can be distributed over the entire network. This method has the constraint that every member of this group can reach all other members in the group via a multi-hop route using only group members.

2.1.9 Conclusions

When designing a new multicast protocol for MANETs a large number of design decisions has to be taken. Some depend on the technical requirements, like the question of whether a position-based or a topology-based protocol should be applied. Others depend on the intended usage scenario. Concluding this classification section, the most important design criteria are the following:

Position-based vs. topology-based: Positions are only available with the existence of a positioning service. Position-based protocols also require location services. But usually the use of additional information yields a more effective packet forwarding.

Tree vs. mesh structure: Trees are better suited for less dynamic or even static networks where tree rebuilds are rarely necessary.

Centralized vs. decentralized organization: A centralized approach pays off, e. g., if the core is able to filter messages and can thus save bandwidth for the receivers. However, a centralized protocol suffers from a single point of failure and may induce high repair costs.

Sender vs. receiver initiation: A sender initiation is better suited if nodes only sporadically want to send data packets. Vice versa, in the case of permanently transmitting senders, the protocol should act in a receiver initiated manner.

Hard state vs. soft state: Similar to the distribution structure, this criterion depends mainly on the dynamics in the network. Hard state approaches only qualify for mainly static networks.

Proactive vs. reactive maintenance: Proactive protocol elements are able to help save maintenance bandwidth by keeping state up-to-date permanently, especially when there is traffic to be routed on a regular basis.

Use of flooding: Flooding should generally be avoided or, at least, kept at a minimum. However, sometimes there is no other way to find a specific node.

2.2 Overview of Existing Multicast Routing Protocols

In this section, we present a selection of the most important existing protocol proposals for multicast in MANETs in order to give an insight in their designs. We briefly describe their algorithms and classify them according to the characteristics described in the previous section (2.1). During the recent years, a large number of proposals have been made. We selected the algorithms described in the following such that they represent a good coverage of the mechanisms ad-hoc multicast protocols are built on. Table 2.1 gives an overview of the described protocols and some more and lists their classifying properties.

Several surveys on multicast routing protocols for MANETs exist: [167] (AMRoute, ODMRP, AMRIS, CAMP, IMAHN), [56] (AMRoute, AMRIS, MAODV, LAM, LGT, ODMRP, CAMP, DDM, FGMP, MCEDAR), and [170] (in German; MAODV, AMRIS, MZR, AMRoute, ADMR, FGMP, ODMRP, NSMP, CAMP). Performance comparisons have been published in [124] (AMRoute, ODMRP, AMRIS, CAMP), [115] (MAODV, ODMRP), and [139] (ADMR, ODMRP, SRMP).

2.2.1 Topology-based Protocols

Tree-based Protocols

ST-WIM Motivated by the PIM sparse mode protocol for wired networks [61], the Shared Tree Wireless Multicast protocol (ST-WIM) [47] was one of the first multicast routing protocols for wireless networks. The data are disseminated over a group-shared tree which is rooted at a rendezvous point. To join a group, a receiver sends a *JOIN_REQUEST* to the rendezvous point via unicast. The nodes in between remember that the request has passed and mark the corresponding link as a downstream link of the multicast group. Depending on the chosen scheme, the node memberships are periodically refreshed (soft state), or the nodes stay members until they decide to send a *QUIT_REQUEST* to the rendezvous point (hard state).
With soft state no specific tree repair strategy is needed as the nodes periodically rejoin and thus refresh or build new tree branches. In the case of hard-state tree maintenance, there are two possibilities to accomplish a tree reconstruction: either the downstream node which has lost the connectivity rejoins its branch by sending a new *JOIN_REQUEST*, or it sends a *FLUSH_TREE* down the branch and the members downstream rejoin by themselves.
The authors assume a clustered multihop network structure in which only clusterheads or gateway nodes take part in supporting the multicast tree while leaf nodes may only join as members via their appropriate clusterhead node.

AMRoute The Ad hoc Multicast Routing protocol (AMRoute) [128, 196] uses unicast tunnels to connect multicast group members via a group shared tree. Before constructing the data deliver tree, a mesh is created that connects all nodes that are either sender or receiver of a group. Initially, each group node represents a mesh consisting of only one node of which it is the core node. A core node periodically sends *JOIN_REQ* messages with increasing TTL values and performs an expanding ring search (ERS) to discovers other members. A merge between two meshes is done by setting up a tunnel link between the the answering node from the other mesh and the requesting core node or any other mesh node on the path. One of the two core nodes in the combined mesh is selected as the new core.
To construct a tree the core node of the mesh periodically sends *TREE_CREATE* messages over the tunnels in the mesh. A mesh node receiving such a message relays it to all links but the one where the message came from. If a node receives the same *TREE_CREATE* message for the second time, it discards the packet and responds with a *TREE_CREATE_NAK* message. The corresponding tunnel is removed from the tree. This process sets up the tree.

MAODV The Multicast Ad-hoc On-Demand Distance Vector (MAODV) protocol [161] is based on the Ad-hoc On-Demand Distance Vector (AODV) routing protocol [152, 151]. Its unicast routing basically operates as follows: A node willing to send data to a destination node checks whether it has a routing entry in its table. If so, the packet is sent to this node. If not, a *Route Request* (RREQ) is sent out via flooding. Any node receiving this RREQ checks whether it is the requested destination or whether it has an entry for this destination. If one of these applies, it generates a *Route Reply* (RREP) message, directed back to the requesting node. The RREP is then forwarded back to the requesting node using the information collected while flooding the RREQ. Each node forwarding the RREP also adds an entry to its routing table, building up the originally requested route.
In multicast mode, a node that wishes to send data to or receive data from a group generates a RREQ message. Senders and receivers are handled dif-

ferently: A sender simply broadcasts a RREQ indicating the group it wants to send data to. The first node that is part of the tree or has a route towards the tree responds to this message, setting up a reverse route to the sender node—similar to the unicast case. A receiver, however, requires more action. It would want to join the tree in order to receive data from it. For this purpose it sets the so-called *J_flag* in the generated RREQ. Such join requests are answered by any nodes that are already part of the multicast tree for the required group. While these responses may generate multiple routes from the tree to the receiver, only one of them is then selected by means of a newly introduced message type: the *Multicast Activation* (MACT) message. This message may also be used to leave a group, pruning the leaf path to the tree.
For maintenance, MAODV follows a centralized approach: The first member of a group will become the group leader responsible for sending out periodic maintenance messages that allow the detection of link breakages within the tree or the detection of partition merges.

AMRIS AMRIS (Ad hoc Multicast Routing Protocol Utilizing Increasing ID-numbers) [194] uses so-called multicast session IDs to build group shared trees for the data delivery. The protocol works in two phases: During *Tree Initialization* one of the senders of a group chooses an ID and broadcasts a *NEW-SESSION* packet containing that ID. On the first reception, the neighboring nodes choose an ID that is higher than the one contained in the packet, leaving some space for nodes joining the network at a later time, and rebroadcast the message. By means of regular beacon messages, the nodes maintain a neighbor table containing the IDs of all neighboring nodes. When a node wants to join the multicast group, it picks one of the neighbors that have a lower ID than the node itself and contacts it with a *JOIN-REQ* message. If that node is not already part of the tree it will also select a parent node to propagate the *JOIN-REQ*.
The second phase is called *Tree Maintenance*. During that phase broken links are handled in two way: (a) if possible, a child node having lost the link to its parent simply rejoins like during the initialization phase; (b) if there is

no potential parent node, the child node broadcasts a *JOIN-REQ* which is flooded over a limited number of hops to search for a new parent node.

Mesh-based Protocols

FGMP The Forwarding Group Multicast Protocol [48] was one of the first mesh-based solutions for multicast and simultaneously introduced the flooding of control packets instead of data packets. It offers two variants for the advertising of a multicast group, one of which is initiated by a receiver (FGMP-RA) and the other one by a source (FGMP-SA). Depending on the chosen scheme, either receivers or senders broadcast their membership information to the entire network. Whichever number of nodes is smaller, that of the sources or that of the receivers, it reduces the control overhead to choose the corresponding advertisement strategy.
For the maintenance of forwarding and multicast group membership information, a soft-state approach is used. A forwarding group node is deleted from this group after timeout. The selection of forwarding nodes is made through the exchange of forwarding (in case of FGMP-RA) oder joining tables (for FGMP-SA) which are built on-the-fly and contain the next hop information for a specific multicast group. Forwarding tables are addressed to the sources of the group, and joining tables to the receivers. The nodes that forwarded one of these messages consider themselves to be part of the forwarding group building the mesh.

ODMRP The On-Demand Multicast Routing Protocol (ODMRP) [122] is an approach based on meshes which gained most respect within the research community in recent years. ODMRP can be seen as the successor to FGMP (see above). It uses soft-state information to manage forwarding and multicast group memberships. Control packets, which optionally can contain data payload, are periodically broadcast through the whole network. The protocol has an extension allowing to exploit position information (if available) to predict node mobility. A distinctive feature is that the proto-

col can be also used for unicast routing, thus making an additional protocol unnecessary.
The building of a new multicast mesh is initiated by the source. A node wanting to send data to a multicast group periodically creates *Join Request* messages. These are broadcast to all nodes within the ad-hoc network in order to advertise a multicast group.
While forwarding such a *Join Request*, the nodes keep track of the nearest upstream node to a source from which the first copy of the request was received using a *Routing Table*. When a multicast group member receives a *Join Request*, it updates the entry according to the source in its *Member Table*. As long as a node has entries in its *Member Table*, it periodically broadcasts a *Join Table* message containing the upstream nodes which were stored in the *Routing Table*. A neighbor whose ID is listed in this message considers itself a member of the forwarding group, adds an entry to its *Forwarding Group Table*, and broadcasts its own *Join Table* to the neighbors. This way, the *Join Table*s construct the shortest path routes from each member to the multicast source, which altogether form a mesh.

PatchODMRP PatchODMRP [120] adds a mesh reconstruction feature to ODMRP (see above). To accomplish this, the forwarding group nodes build up neighbor tables by sending MAC layer beacons, and watch the availability of their upstream neighbors. In case an upstream link breaks, the forwarding node floods an *ADVT* packet with a small time-to-live value to search for other possible upstream neighbors. Nodes which forward this advertisement add a corresponding entry to their routing table so that they can forward a potential answer to the request.
If the *ADVT* arrives at a forwarding node which serves the requested group and source and which is nearer to the source than the sender of the *ADVT*, the node replies by sending a *PATCH* packet. The intermediate nodes are now temporary forwarding group nodes until either the next ODMRP *Join Table* from a receiver passes, or the patched route expires because the initiator decides to use a more favorable *PATCH* packet it got.

NSMP The principle functionality of the Neighbor Supporting ad-hoc Multicast Protocol (NSMP) [121] for multicast mesh creation is identical to that of ODMRP (see above). A source that has data to send announces itself once via a network broadcast (*FLOOD_REQ*), initiating reply packets (*REP*) from the interested receivers. The *REP* packets build up a forwarding group which in turn will be responsible for the data delivery. An additional class of nodes introduced by NSMP are the *neighbor* nodes of the mesh, which do not belong to the mesh themselves but know that there is a mesh node within their transmission range.
Once the source has established an initial mesh, it starts to periodically send out *LOCAL_REQ* packets. The difference between *FLOOD_REQ* and *LOCAL_REQ* packets is that the latter are forwarded only by mesh and neighbor nodes. Thus only nodes located at most two hops away from the multicast mesh know about the source. If a node further away wants to join the group, it floods a *MEM_REQ* message, using an expanding ring search (ERS) starting at three hops. The new receiver will probably get more than one route discovery packet in answer to his request, and subsequently it chooses the shortest path containing the most forwarding group nodes to minimize the expansion of the mesh.
A group leader is chosen from among all the sources within each group. As every source also receives the *LOCAL_REQ*'s of its colleagues, they all have the same notion of sources and select the one with the lowest ID as their leader. This special node periodically broadcasts *FLOOD_REQ* messages which help to recover network partitions.

DCMP The Dynamic Core based Multicast routing Protocol (DCMP) [55] was also modeled on ODMRP (see above). The main difference from ODMRP is that DCMP classifies sources into *active sources*, *core active sources* and *passive sources*. Active sources are sources as known from ODMRP, and core active sources send out *JoinReq*'s on behalf of their passive sources. In this way control overhead is reduced.
The classification is done as follows: Initially, all sources are active, i. e., they send out *JoinReq* messages to the whole network that contain a pa-

rameterized counter of how many passive sources they are able to support. If an active source receives such a *JoinReq*, it checks three conditions to decide whether to request the passive status. First, the received *JoinReq* must contain a *CoreAcceptance flag* which is set if the originating source is able to support additional passive sources; second, the originating source has to be at most as far as two hops away; and last, its ID has to be smaller the the one of the other source. On conformance to all three requirements, it sends a *PassReq* message to the desired core, while a locking mechanism prevents that two of these operations constrain each other.

CAMP The Core-Assisted Mesh Protocol (CAMP) [74] was the first mesh-based protocol not to use periodic flooding of control packets in the whole network. Instead, a node willing to join a group uses an expanding ring search to reach one of the current members of that group. It then chooses one of the reverse paths constructed through replies from different members. This way, the nearest path from the joining node to the mesh is added to the mesh.

To avoid even this kind of flooding, CAMP may use an arbitrary number of cores, which are responsible for a certain group and which can be contacted directly for a join. To operate correctly, the protocol relies on the existence of a unicast routing protocol which finds the shortest path to every destination. Once built up, the mesh of CAMP provides the shortest paths from each sender to each multicast receiver. This property is ensured by a mechanism that works as follows: If a destination node has not received any data packet on the shortest path for a certain time, it starts sending out *heartbeat* messages across the next hop on the last known shortest path. These *heartbeat* messages are forwarded as long as all the hops already belong to the mesh, and they are converted into *push join*s if there is a 'mesh gap' on the shortest path.

Each receiver selects at least one of its neighboring nodes as an *anchor* to the group and announces its choice to them. This ensures that no mesh member which is required for the maintenance decides to leave the group.

In the event that a node wants to send data to a group without receiving or forwarding the packets from other sources within this group, it can join as a *simplex* instead of as a *duplex* member. Thus, the node will become part of the mesh but no packets will be delivered into its mesh branch.
Mesh partitions are handled by the cores through a periodic exchange of messages among each other. Nodes that are forwarding these messages and that are not a member of the mesh at the same time, are forced to join. Thus two mesh partitions will eventually merge.

MCEDAR The MCEDAR (Multicast Core-Extraction Distributed Ad-Hoc Routing) protocol [168] is an extension to the CEDAR unicast protocol [169] by the same authors. CEDAR provides a *core* of the network and specifies a corresponding core broadcast mechanism which uses unicast messages between the core members to distribute messages. For its multicast functionality, MCEDAR builds a mesh which is called *mgraph* and which consists of a subgraph of the network core. The multicast forwarding is performed by flooding data packets within the *mgraph*.
If a node wants to join the multicast group, it depends on whether it belongs to the CEDAR core or not. In CEDAR, nodes that themselves do not belong to the core have a core node in their one-hop neighborhood, the so-called dominating core node. Only core nodes can join the *mgraph* directly, the other nodes have to ask their dominating core node to join for them. During the join process, each *mgraph* node is assigned a *JoinID* which represents the order in which the nodes joined the multicast group. They are used to prevent loop formation during mesh reconstructions such that only nodes with smaller IDs search for nodes with higher IDs.
A robustness factor R determines how many parents each node should have within the *mgraph* to ensure a certain degree of connectivity and robustness against mobility.

Protocol name	version	Unicast-dependent	Unicast-capable	Structure	Use of positions	State	Flooding control	Flooding data	Flooding when	Flooding type	Initiation
RBM [51, 50]		• UNIRP [52]	–	RP's/source specific	–	combined					receiver
ST-WIM [47]		○	–	shared tree	–	alternatively	–	–	–	–	receiver
DSR-ext [136]	DSR [101]	•	–	source tree	–	–	•	–	routes	broadcast	source
LAM [97]	TORA [148]	•	–	shared tree	–	hard	–	–	–	–	receiver
hyper-flooding [144]		–	–	mesh	–	–	–	•	all data	hyper	source
FGMP [48]		–	–	mesh	–	soft	•	–	periodically	membership	either
AMRoute [128, 196]		○	–	shared tree	–	combined	•	–	periodically	ERS	either
LBM [113, 114]	LAR[112]	–	–	geocast	forwarding zone	–	–	•	all data	geographical	source
reliable [78]		○	–	core based tree	–	hard	•	–	rebuild	scoped	source
CAMP [74]		○	–	mesh	–	combined	•	–	join	ERS	receiver
MAODV [161]	AODV [152]	–	•	shared tree	–	soft	•	–	periodically	broadcast	either
ODMRP [122]		–	•	mesh	mobility prediction	soft	•	○	periodically	broadcast	source
MCEDAR [168]	CEDAR [169]	•	–	mesh	–	hard	•	–	periodically	beacons	either
BEM[146, 147]		–	–	source tree	–	combined	•	•	join rebuild	broadcast scoped	receiver
AMRIS [194]		○	–	shared tree	–	soft	•	–	init rebuild	broadcast scoped	source
CGM [127]		–	–	shared tree	–	soft	•	–	join	scoped	receiver
CBM [198]		–	–	geocast	mobility prediction	–	–	•	all data	restricted	source
NSMP [121]		–	–	mesh	–	soft	•	–	periodically join	broadcast ERS	source
Role-Based [37]		–	–	geocast	multicast group	–	–	•	all data	restricted	source
ABAM [180]	ABR [179]	–	–	source tree	–	hard	•	–	init rebuild	broadcast scoped	source
OLAM [21]/DSM [20]		–	–	source tree	topology creation	–	•	–	periodically	broadcast	source
PatchODMRP [120]		–	•	mesh	–	soft	•	○	periodically	broadcast	source
MZR [62]		–	–	source tree	–	soft	•	–	periodically rebuild	scoped ERS	source
simple [94]	DSR [101]	–	–	broadcast	–	stateless	–	•	all data	broadcast	source
DDM [99]		○	–	source tree	–	stateless/soft	–	–	–	–	source
ADMR [95]		–	–	source tree	–	soft	•	•	join periodically	broadcast	source
SRMP [138, 140]	DSR [101]	–	–	mesh	–	soft	•	–	join	broadcast	source
DCMP [55]		–	–	mesh	–	hard	•	–	periodically	broadcast	source
LGT [43]		–	–	source tree	tree construction	soft	•	–	periodically	membership	–

Table 2.1: *Overview of proposals for ad-hoc multicast in chronological order*
(Key: ○ for unicast-dependent: may be based on any arbitrary unicast routing protocol, for data flooding: data may optionally be flooded)

2.2.2 Position-based Protocols

OLAM / DSM Basagni et al. describe a multicast protocol that uses the positions of the nodes in the network. In its first version it was called On-Demand Location Aware Multicast (OLAM) [21], later the name was changed into Dynamic Source Multicast (DSM) [20] (probably following Dynamic Source Routing (DSR) [101]). However, both papers basically describe the same algorithms. The principle of the protocol is that all nodes are aware of their position and all the positions of the other nodes in the network. I. e., each node that wants to send data to a multicast group must have a complete snapshot of the nodes' positions in the network. Also, it is assumed that it knows about the group membership information for its destination group. It then locally calculates a geographical Steiner tree [86] spanning all group members. This tree is encoded using a Prüfer sequence [153] and is added to every multicast packet. The advantage of such a sequence is that the length of the code is limited to $n - 2$, where n is the number of nodes in the tree. Based on the information contained in the packet, all intermediate nodes are then able to forward the packet to all group members. Thus, OLAM/DSM implement a geographical multicast source routing.

Location Guided Tree Construction A very similar approach was proposed by Chen and Nahrstedt in [43]. In contrast to the DSM protocol described in the paragraph above, Chen and Nahrstedt additionally propose a mechanism to disseminate the locations of all group members to each other. This mechanism follows a proactive approach in the sense that it forces nodes to generate packets addressed to the multicast group even if they do not have any data to send. Every packet contains the geographic location of the sending node. Thus, all group members are constantly kept aware of the other group members' positions. If a node stops transmitting these position updates, it is removed from the group, applying a soft-state procedure. Still missing is a possibility to join a group. This is assumed to be handled externally by some application level mechanism.

For the construction of multicast trees out of the geographic locations of the destination nodes the authors investigate two different methods. The first one, location-guided k-ary tree (LGK) construction, tries to build trees with a degree of k, a predefined parameter. The second one uses location-guided Steiner trees (LGS) in order to build minimum spanning trees, similar to DSM (see above). The comparison of these two methods reveals that LGS trees provide a low-bandwidth packet distribution if position information is up-to-date. LGK trees, however, have a much lower computational complexity and distribution delay, even with not so up-to-date positions.

2.3 Conclusions

There are many different protocol proposals for multicast in MANETs that vary according to a number of classification characteristics. During the past few years, nearly all possible combinations of design decisions have been investigated, and their performance is now well documented. However, the proposed protocols are mostly topology-based. Only very few approaches to position-based multicast were designed. What is more, these approaches only allow multicast in small groups using a tree that is predetermined by the sender. Thus, these protocols are inflexible with respect to mobility.

The protocols presented in this thesis will fill this gap, allowing multicast for scalable group sizes in networks with high mobility. Starting from a generalization of greedy position-based unicast routing, the remaining issues are identified and solved step by step, finally resulting in a position-based multicast protocol for scalable group sizes that allows for highly mobile nodes.

CHAPTER 3

Position-Based Multicast

In this chapter we present Position-Based Multicast (PBM) [119, 132, 134, 133], a first approach to *position-based* multicast routing for MANETs. It was developed by Lang et al. and represents an important step towards the scalable protocols that we will describe in the following chapters. PBM neither requires the maintenance of a distribution structure (e. g., a tree or a mesh) nor resorts to flooding of data packets. Instead, a forwarding node uses information about the positions of the destinations and its own neighbors to determine the next hops that the packet should be forwarded to. It is thus very well suited for highly dynamic networks. PBM is a generalization of existing position-based unicast routing protocols such as face-2 [36] or GPSR [106]. The key contributions of PBM are rules for the splitting of a multicast packet's path and a repair strategy for situations where no direct neighbor exists that makes progress toward one or more destinations. The characteristics of PBM are evaluated in detail by means of simulation.

3.1 Introduction

Position-based routing can be divided into two main functional elements: the location service and position-based forwarding. The *location service* is used to map the unique identifier (such as an IP address) of a node to its geographical position. For the remainder of this chapter we assume that an appropriate location service is present which supplies the sender of a packet with the geographical position of the packets' destinations. Examples of existing location services that can be used for this purpose are Homezone [75], the Grid Location Service (GLS) [126] or the location service part of DREAM [22].

Position-based forwarding for unicast is performed by selecting one of the forwarding node's neighbors in transmission range as the next hop the packet should be forwarded to. Usually, for the forwarding decision, the geographical positions of the node itself, its direct neighbors, and the packet's destination need to be known. With this information, the forwarding node selects one of its neighbors as a next hop such that the packet makes progress towards the geographical position of the destination.

It is possible that there is no neighbor with progress towards the destination although a valid route to the destination exists. The packet is then said to have reached a local optimum. In this case, a *recovery strategy* is used to escape the local optimum and to find a path towards the destination.

The most important characteristic of position-based routing is that forwarding decisions are based on *local knowledge*. It is not necessary to create and maintain a global route from the sender to the destination. Therefore, position-based routing is commonly regarded as being highly scalable and very robust against frequent topological changes. It is particularly well suited for environments where the nodes have access to their geographical position, such as in inter-vehicle communication [137, 191, 67].

In order to extend position-based routing to multicast, two key problems have to be solved. First, at certain nodes, the path of a multicast packet has to be split into multiple directions in order to reach all destinations[1]. The challenge here is to decide when such a node is reached and a copy of the packet should be created. Second, the recovery strategy used to escape from a local optimum needs to be adapted to take multiple destinations into account. The key contributions of this chapter are solutions to both problems. The proposed algorithms are evaluated by means of simulation.

3.2 The Position-Based Multicast Algorithm

For multicast it is necessary to establish a distribution tree among the nodes along which packets are forwarded towards the destinations. At the branching points of the tree, copies of the packet are sent along all the branches. Two—potentially conflicting—properties are desirable for such a distribution tree: (1) The length of the paths to the individual destinations should be minimal and (2) the total number of hops needed to forward the packet to all destinations should be as small as possible. If the topology of the network is known, a distribution tree that optimizes the first criterion can be obtained by combining the shortest paths to the destinations. Wherever these paths split, the packet is duplicated. The second criterion is optimized by means of Steiner trees (see e. g., [86]) which connect the source and the destinations with the minimum number of hops. A formulation of the Steiner problem for wireless networks where packets are broadcast to neighboring nodes is given in [193]. It has also been investigated in DSM [20] and LGT [43] (see Section 2.2.2). However, with greedy position-based routing, routing decisions are based solely on local knowledge; thus neither the shortest paths to all destinations nor (heuristics for) Steiner trees can be used directly. Instead, PBM uses locally available information to approximate the optima for both properties.

For the remainder of this work we assume that each node that forwards a packet has access to the following information:

[1]In the following, we will use the term "to split a packet" whenever the destinations in the packet are divided into multiple sets, the data payload is copied into as many new packets, and the packets are sent out to different next hops.

1. *The node's own geographic position*: This information can be provided by a positioning service such as GPS [105] or WLAN-based positioning [84, 109].

2. *The position of all neighbors within transmission range*: The position of a node is made available to its direct neighbors in the form of periodically transmitted beacons.

3. *The positions of the destinations*: These may be included in the packet or available locally (i. e., because a location service distributes position information about all nodes to all other nodes within the network, such as in DREAM [22]).

Given this information, the main task of a forwarding node in PBM is to find a set of neighbors that should forward the packet next. We call these neighbors the *next hop nodes*. The current node will assign each destination of the packet to exactly one next hop node. Each next hop node then becomes the forwarding node for this packet towards the assigned destinations. If the current node selects more than one next hop node, then the multicast packet is split. This may be required in order to reach destinations which are located in different directions relative to the forwarding node. The most important property of PBM is that each forwarding node autonomously decides how to forward the packet. This decision requires no global distribution structure such as a pre-established tree or mesh.
There are two distinct cases that can occur when a forwarding node selects the next hop nodes: Either for each destination there exists at least one neighbor which is closer to that destination than the forwarding node itself, in which *greedy multicast forwarding* is used. Otherwise, the node employs *perimeter multicast forwarding*.

3.2.1 Greedy Multicast Forwarding

As discussed above, a multicast distribution tree ideally optimizes two criteria. First, the distance towards the destination nodes should be minimized and hence the progress of the packet towards the destinations maximized. Second, the (global) bandwidth usage should be minimized. Thus the objective function of a forwarding node should consist of two elements, one for each objective. Optimizing the progress of the packet can be done in the following way. Let k be the forwarding node, N the set of all neighbors of k, W the set of all subsets of N, Z the set of all destination nodes, and $d(x, y)$ a function which measures the distance between nodes x and y. Given a set of next hop nodes $w \in W$, the overall remaining distance to all destinations of a multicast packet can be calculated as shown in the following equation:

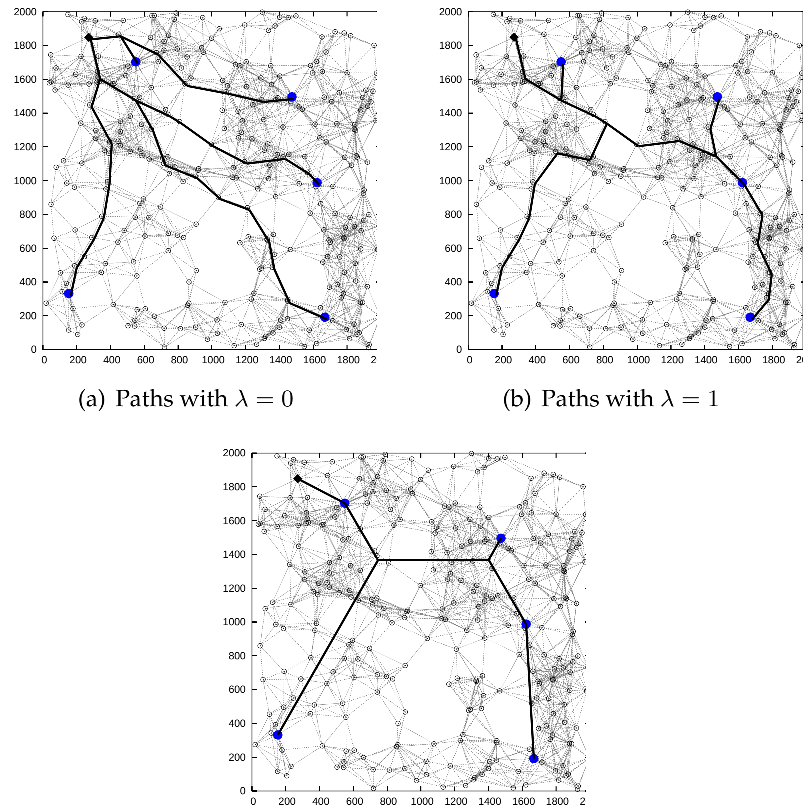
$$f_d(w) = \sum_{z \in Z} \min_{m \in w} (d(m, z)). \tag{3.1}$$

In this equation, for each destination the next hop node in the set w is chosen which is closest to that destination. Using Equation 3.1 as the sole optimization criterion would lead to a splitting of the multicast packet as soon as there is no single neighbor which provides the greatest progress towards all destinations. This may be undesirable since it ignores the bandwidth usage.

In order to consider the bandwidth usage, we include the number of next hop nodes as a second element in the optimization criterion. The overall optimization criterion that determines which set of next hop nodes $w \in W$ should be selected as the next forwarding nodes is then given as:

$$f(w) = \lambda \frac{|w|}{|N|} + (1 - \lambda) \frac{\sum_{z \in Z} \min_{m \in w}(d(m, z))}{\sum_{z \in Z}(d(k, z))}. \tag{3.2}$$

The first part of the equation determines the number of next hop neighbors and normalizes it to a value between $[0, 1]$ by dividing it by the total number of neighbors of k. The second part determines the remaining overall distance from the next hop nodes to the destinations and normalizes this to a value between $[0, 1]$ by dividing it by the remaining overall distance calculated from the forwarding node k to the destinations. $\lambda \in [0, 1]$ determines the weight of each objective. If λ is close to 0, multicast packets will be split early, while if λ is close to 1, the multicast packet will only be



(a) Paths with $\lambda = 0$

(b) Paths with $\lambda = 1$

(c) Steiner tree (ignoring intermediate nodes' positions and reachability)

Figure 3.1: *Effect of parameter λ on the multicast paths*

split if this is enforced by the restriction that there must be progress at each destination. An example for the impact of λ on the path that a multicast packet takes through the network is shown in Figure 3.1.

It can be expected that the number of hops that a packet traverses from the source to a given destination increases with increasing λ, i. e., the path towards each destination becomes less direct. On the other hand, the total number of single-hop transmissions required to deliver the packet from the source to all destinations is likely to decrease when λ increases from 0 up to a certain value $s < 1$. The decrease of single-hop transmissions when λ is increased from a value close to 0 to s is caused by the fact that packets are split later, thus less single-hop transmissions are needed. However, if a packet is split very late, i. e., $\lambda > s$, then the total number of hops may increase again.

These considerations can be illustrated by the simple topology given in Figure 3.2. Let A be the forwarding node and B and C the destinations of a multicast packet. Let us further assume that the node density is high enough that a packet can be split virtually anywhere and that the distance to the destinations is much greater than the radio range. If A decided to split the packet, the copies would be forwarded along $\overline{AB}$ and $\overline{AC}$, taking the most direct path to the destinations. To keep the number of total hops needed to distribute the packet to all destinations to a minimum, the packet should be forwarded along $\overline{AD}$, and node D should then split the packet and send copies to the final destinations (as indicated by "Steiner" in the graph). If the packet is not split at D but forwarded further, the total number of hops, as well as the lengths of the individual paths to the destinations, increase again. Therefore, the packet should ideally be split somewhere between A and D. Since λ determines how early a packet should be split, there will be a value $s < 1$ for λ, where the total number of single-hop transmissions will be minimal. We determine a value for s by means of simulation in Section 3.3.

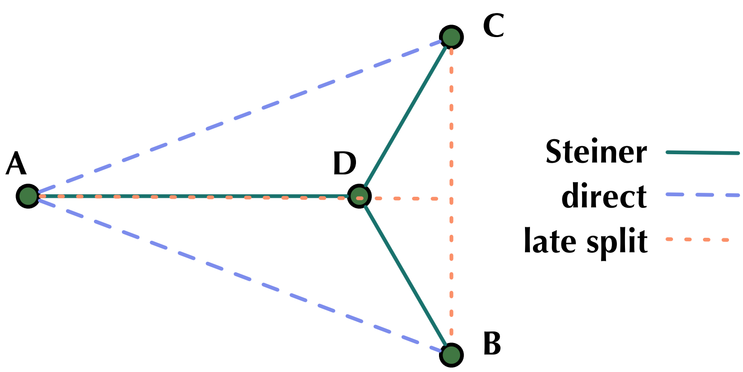


Figure 3.2: *Effect of λ on the number of single-hop transmissions*

3.2.2 Perimeter Multicast Forwarding

As mentioned before, applying greedy multicast forwarding may lead to a situation where the packet arrives at a node that does not have neighbors providing progress to one or more destinations. An example of this is depicted in Figure 3.3: The copy of the multicast packet which is on its way to D_2, D_3, and D_4, as well as the copy for D_5, gets stuck in a local optimum.

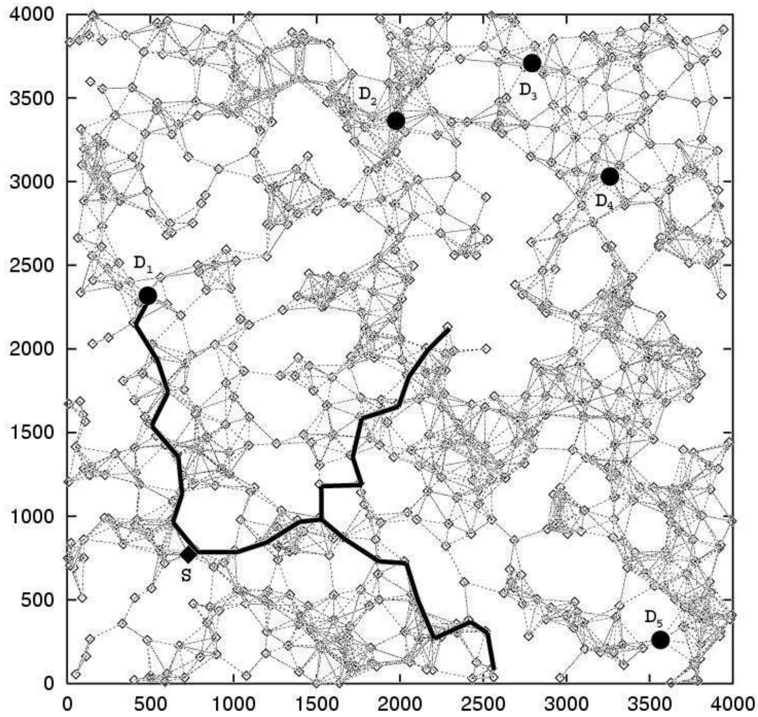


Figure 3.3: *Greedy Multicast Routing Failure*

For position-based unicast, this problem has been solved by applying a modification of the right-hand rule ([36, 106]). The basic idea is to traverse the boundaries of gaps in the network until greedy forwarding can be resumed. To this end, the graph formed by the connections (edges) between mobile nodes is planarized, i. e., intersecting edges are removed. This planarization is based on Relative Neighborhood, or Gabriel Graphs [181, 73].

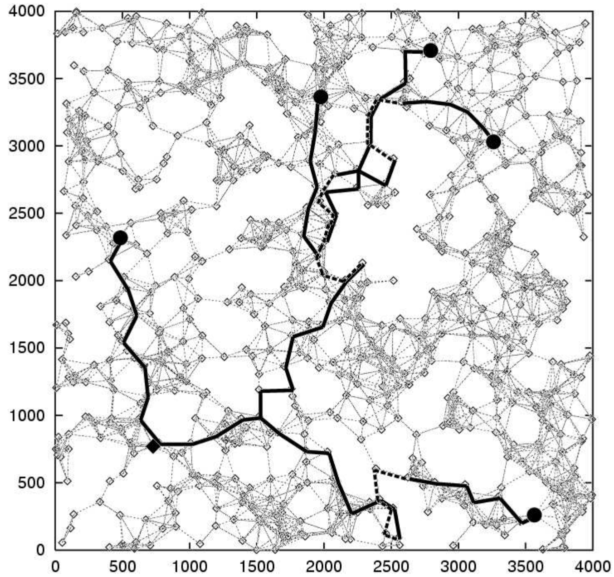
It can be done individually by each node based on local knowledge and does not partition the graph.
On the planarized graph the right-hand rule can be used to escape the local optimum: The node where the local optimum is reached calculates a virtual edge from itself to the destination. The packet is then transmitted over the next edge counter-clockwise to that virtual edge. A packet transmitted in this way is said to be in *perimeter mode*. When a packet is received by a node in perimeter mode, then this node checks whether it is closer to the destination than the node at which the packet entered perimeter mode. If this is the case, the packet is reverted to *greedy mode* and forwarded in greedy fashion. If this is not the case the packet is forwarded over the next edge counter-clockwise to the edge it arrived on. The combination of perimeter and greedy forwarding guarantees that the destination is reached, as long as the network is static and as long as a valid connection between source and destination exists.
For PBM we generalized this algorithm to support packets with multiple destinations. If a node in PBM detects that it has no neighbors with forward progress to one or more destinations, multicast perimeter mode is initiated for these destinations, while for all other destinations, greedy multicast forwarding is used. As in the unicast case, the parameter mode is performed on the planarized graph (PBM uses Gabriel Graphs for planarization). The virtual edge used for the initialization is calculated as the connection between the current node and the position representing the average of the positions of the affected destination nodes. The average of a number of positions can be obtained by calculating the arithmetic mean of each coordinate. The multicast perimeter packet is then transmitted over the first edge counter-clockwise to the virtual edge.
When a node receives a perimeter multicast packet, it checks for each destination whether it is closer to that destination than the node at which the packet entered perimeter multicast mode. For all destinations where this is the case, greedy multicast forwarding can be resumed; for all other destinations, perimeter multicasting is continued by transmitting the packet over the next edge counter-clockwise to the edge on which the packet arrived.

Automatically splitting a packet into copies that are to be forwarded in greedy multicast mode and a copy that is to use perimeter multicast mode may cause the transmission of the same packet to two nodes which are located in the same direction, or even to the same node twice. In order to reduce the load on the network, PBM includes an optional combination of greedy and perimeter multicast forwarding: If some, but not all, destinations of a packet require perimeter multicast forwarding, then the next hop is determined using the perimeter rules listed above. All copies of the packet with destinations for which greedy forwarding could be used also select this node as the next hop if it provides progress towards the copy's destination. This reduces the number of copies of the same packet circulating in the network. It comes at the cost of a potentially increased path length towards the individual destinations. Figure 3.4 shows how the problem depicted in Figure 3.3 is solved using perimeter multicast routing with and without combining perimeter and greedy packets.

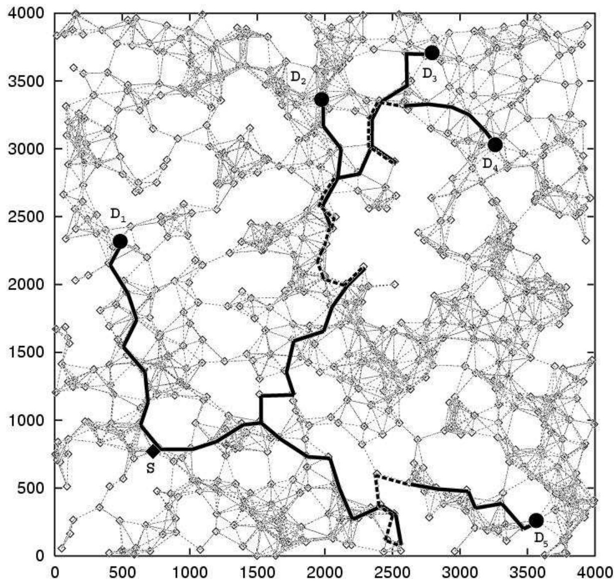
3.3 Evaluation

We evaluated the performance and behavior of PBM by means of simulation. The primary goal of these simulations was to understand the proposed routing algorithm with acceptable empirical significance when used in network topologies of reasonable size. For this purpose, our implementation in the well-known network simulator ns-2 [5] was not suitable for the following reasons:

- The protocol is highly complex. In order to determine the minimum of the objective function given in Equation 3.2, every possible subset of neighbors has to be considered—at each forwarding node, for each packet. This results in an exponential increase in the computational complexity, which depends on both the number of neighboring nodes and the number of destinations in the packet.
- The simulated network should be sufficiently large to be able to distinguish between flooding and multicasting a packet. Given a radio



(a) Paths taken without combining perimeter and greedy packets (solid lines: greedy mode, dashed lines: perimeter mode)



(b) Paths taken combining perimeter and greedy packets (solid lines: greedy mode, dashed lines: perimeter mode)

Figure 3.4: *Perimeter multicast routing*

range of 250 meters and 5 or more destinations, areas with a side length of a few hundred meters are too small for this purpose. Larger areas also require a higher number of nodes. In ns-2, the number of nodes is highly constrained by the available computing power and main memory.

- The early loss of a multicast packet will lead to a loss for multiple destinations. Thus the loss rate is subject to a much higher variance than for unicast. As a consequence, the number of simulation runs should be high enough.

For these reasons, we decided to implement our own simulator in C++ without the use of a dedicated simulation environment. Thus, we were able to simulate networks that contained more than 1,000 nodes on an area of 4,000 m $\times$ 4,000 m or larger, and the number of simulation runs was on the order of 1,000 for a large number of parameter combinations (about 200). In our simulation, nodes can communicate if they are within radio range; the transmission of a packet takes 10 ms, and there is no simulation of a MAC layer. This approach significantly reduces the simulation complexity. We are fully aware that this does not allow the investigation of MAC layer interaction with PBM. However, this approach had the key advantage that we were able to observe the characteristics of the routing algorithm itself in a much more detailed manner and with much more empirical significance than could otherwise be done.

3.3.1 Simulation Setup

We simulated the behavior of PBM using three different simulation areas: small (2,000 m $\times$ 2,000 m), medium (4,000 m $\times$ 4,000 m), and large (8,000 m $\times$ 8,000 m). For each area we investigated multiple node densities (30, 40, 50, 60 nodes per km^2), with the nodes initially being randomly placed in and equally distributed over the simulation area. Node movement follows the Random Waypoint (RWP) mobility model [101, 38]: Each node is assigned a random destination position to which it moves with an assigned speed. The node speed was randomly chosen with an equal distribution for each node out of an interval between 0 m/s and a certain maximum speed (0 m/s, 10 m/s, 20 m/s, 30 m/s, 40 m/s, 50 m/s); the pause time, i. e., the time before a node arriving at its destination is assigned a new destination and speed, was set to 0 seconds. One sender and a number of receivers (5, 10, 15, 20, 25, 30) for each packet were chosen such that all destinations were in the same network partition as the sender. Then one packet was transmitted. After the packet traversed the network, the nodes were redistributed, and a new sender as well as new receivers were selected. This process was repeated 1,000 times.[2]

3.3.2 Delivery Rate

As in position-based unicast, PBM is guaranteed to successfully deliver all packets in a static network where the sender and all receivers reside within the same network partition. In a dynamic network, the use of the perimeter mode may lead to routing loops and thus to packet drops. Figure 3.5 shows how such a loop can come into existence. In this figure, the source of the packet is S, the destination is D, and at node u the packet enters perimeter mode. While the packet traverses the link from v to w, a connection is established between x and v because of node movement. The packet will then be caught in the triangle formed by v, w and x and will consequently be dropped.
We investigated the likelihood of packet loss caused by this type of event with respect to mobility and node density. Only those simulation runs were taken into account where the sender and all receivers resided within the same partition for the complete simulation run. We counted the number of destinations that were not reached and related it to the overall number of destinations. The result constitutes the loss rate. Figure 3.6 shows the loss rate for the medium-size area with five destinations per transmitted

[2] [26, 40, 197, 27] describe some negative properties of the RWP model which apply to scenarios running for a few minutes. E.g., after some time, average mobility decreases and the node distribution is not uniform anymore. However, these effects did not occur in the simulations described here since the scenarios were reset for each single packet.

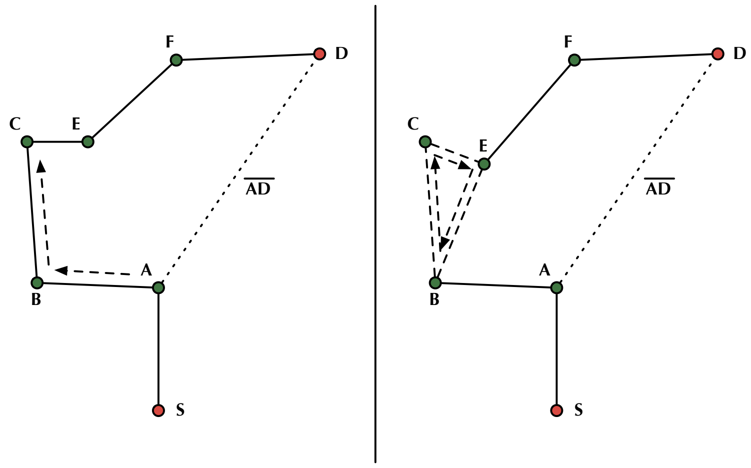


Figure 3.5: *Routing loop in a dynamic network*

packet. It can be seen that the likelihood of a packet drop caused by a routing loop increases as node density decreases. This happens because routing loops can only occur in perimeter mode, and the likelihood of a packet using the perimeter mode increases as node density decreases. Also it can be observed that the likelihood of a routing loop increases if node mobility increases. This is not surprising since node mobility is the reason why a routing loop is formed. Examining the values of the loss rate, it can be noted that it remains fairly low (below 2 %) for node densities above 50 nodes per km^2, even if the node mobility is extremely high.

The second problem that perimeter routing may encounter is caused by the fact that the border of an area without nodes is always traversed counter-clockwise, even though half of the time a clockwise traversal would lead to a shorter route. If the area traversed in this way is very large, or if it is the outer boundary of the network, the required number of hops for this traversal may be unacceptably high if the wrong choice for the orientation of the traversal is made. An example of this is depicted in Figure 3.7.

In order to determine the effect of this problem, we assigned each packet a hop-count. If this hop-count exceeds a predefined value, the packet is dropped. The value was set to 200, which prevents a packet from travers-

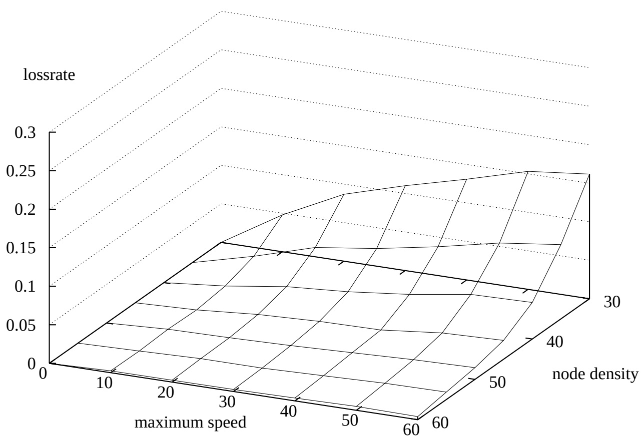


Figure 3.6: *Lossrate without a hop limit*

ing the outer boundary of the network. Any packet exceeding this hop count was dropped. This was done in addition to the packet drops reported above. The result of this simulation is shown in Figure 3.8. It should be pointed out that this figure does include the drops caused by looping packets. Given this fact, it is remarkable that the total number of lost packets is almost completely independent of node speeds. It is easy to see that the likelihood of encountering a perimeter that leads to a traversal of the boundaries of the network depends only on the node density. However, at first glance one would expect that this is in addition to the packet drops caused by routing loops. This is not the case since packets that traverse the boundary of the network have a much longer path than other packets. Thus the likelihood that they will encounter a routing loop is much higher than for other packets. As a consequence, the vast majority of packets that are caught in routing loops are packets that traverse the boundary of the network. An increase in speed therefore only changes the reason to discard a packet (routing loop vs. hop count exceeded) but has no significant impact on the overall loss rate in Figure 3.8.

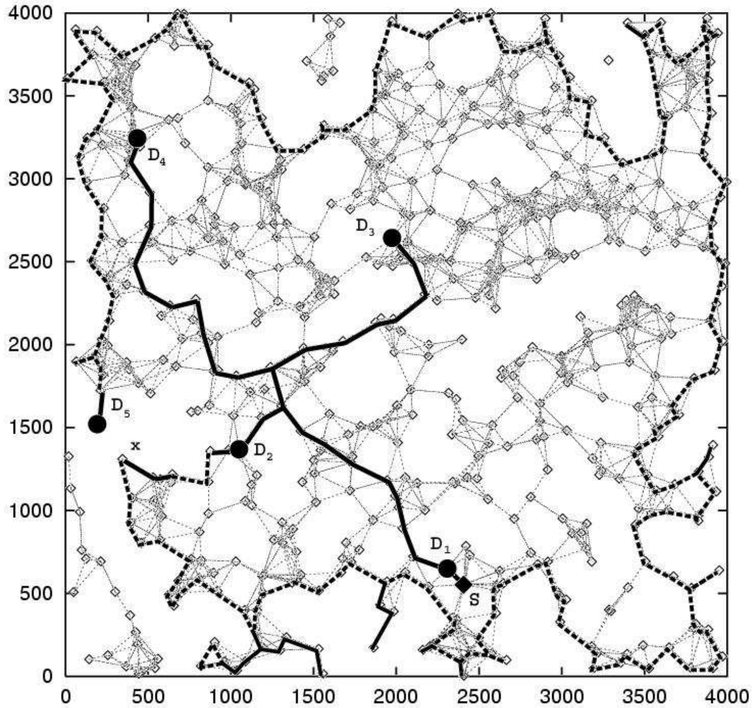


Figure 3.7: *The perimeter problem in a static network (solid lines: greedy mode, dashed lines: perimeter mode)*

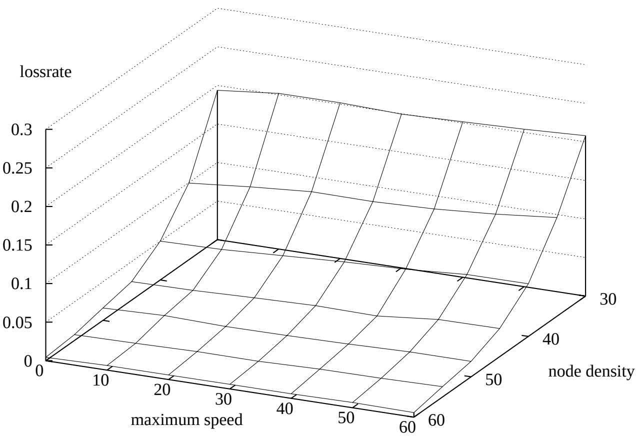


Figure 3.8: *Lossrate with a hop limit of 200*

It can be said in conclusion that for node densities of about 50 nodes or more per km^2, PBM will have very low loss rates. The main cause of packet loss is the traversal of the network boundary. If the network boundary is sufficiently removed from communication partners, the loss rates should decrease substantially. Furthermore it seems worthwhile to investigate approaches to improve the decision about the orientation of the traversal if information about the boundaries of the network (e. g., rivers, lakes) is available. Since PBM is a generalization, these observations also hold for position-based unicast routing as proposed in face-2 and GPSR.

3.3.3 Average Path Length

We define the *average path length* as the average number of hops that a packet traverses on its path from the sender to the receivers. Thus the average path length measures how direct the path towards the destinations is and thereby how much delay the packet will encounter. We were interested in understanding how the choice of λ would influence the average path length. Our hypothesis was that the average path length would increase with increasing λ, since a small value for λ would lead to an optimization of the packet's progress, while a large value for λ would delay the splitting of the packet. We varied λ from 0 to 1 in increments of 0.05 for all combinations of the remaining simulation parameters as described above. We considered only those simulation runs which did not include packet loss or packets which traversed the outer boundary of the network.

Figure 3.9 shows how the path length depends on λ for the medium-size region. This figure contains 24 graphs, each representing one combination of parameters: 40, 50 and 60 nodes per km^2, maximum speed of 0, 20, 40, 60, and the optional combination of greedy and perimeter packets turned on and off. There are three groups of graphs, one group for each node density. It is clear that with an increase in node density the path length will decrease since a more direct path becomes possible. All 24 graphs show the same main characteristic: The path length increases steadily while the value for λ is increased.

Surprisingly, the combination of greedy and perimeter multicast packets did not have a major impact in any of our simulation runs. A further investigation suggested that it rarely alters the path of the packet significantly. As one would expect, the maximum speed of the nodes had no impact on the path length.

3.3.4 Number of Single-Hop Transmissions

The *number of single-hop transmissions* is determined by counting all transmissions that are required to forward the multicast packet to all destinations. It is a measure of the load on the network, caused by the multicast packet. As described in the previous section, we expected that for a given set of parameters the number of single-hop transmissions would reach a global minimum for a value of λ between 0 and 1.

Figure 3.10 shows how the number of single-hop transmissions depends on λ for the medium-size region. Again, the node density has a major impact: the more nodes, the fewer single-hop transmissions are required. This results in the same grouping of graphs as was observed for the path length.

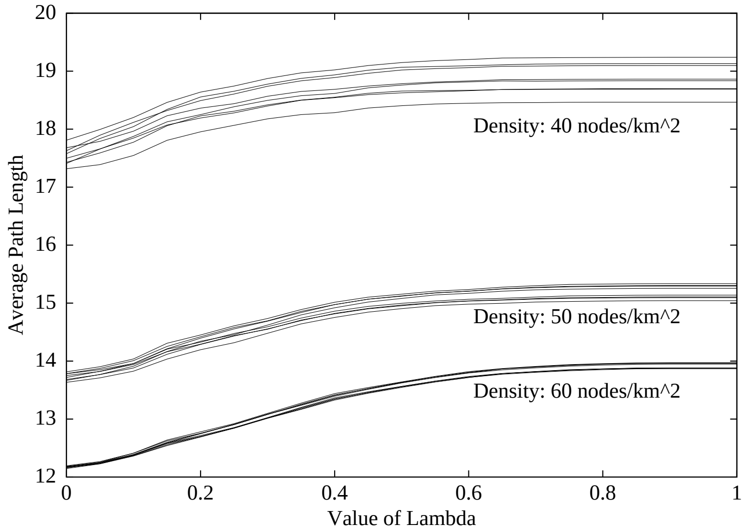


Figure 3.9: *Effect of parameter λ on the average path length*

Over all simulations the shape of the graph is almost identical for all parameter combinations, with the minimum between 0.3 and 0.6. Neither the maximum speed nor the combination of greedy and perimeter packets had a significant impact on the number of single-hop transmissions.
These results indicate that a true trade-off between the goals of minimizing the average path length and minimizing the number of single-hop transmissions exists only for values of λ between 0 and 0.6. Values greater than 0.6 do not lead to a further improvement in the number of single-hop transmissions.

3.3.5 Bandwidth Reduction

Multicast is primarily used to reduce the bandwidth requirement if the same packet needs to be delivered to multiple destinations. Thus it is interesting to compare the number of single-hop transmissions required to transmit the packet when unicast is used to the number required when the packet is delivered via multicast. In order to enable this comparison, we determined the average path length with λ set to 0. We then multiplied this

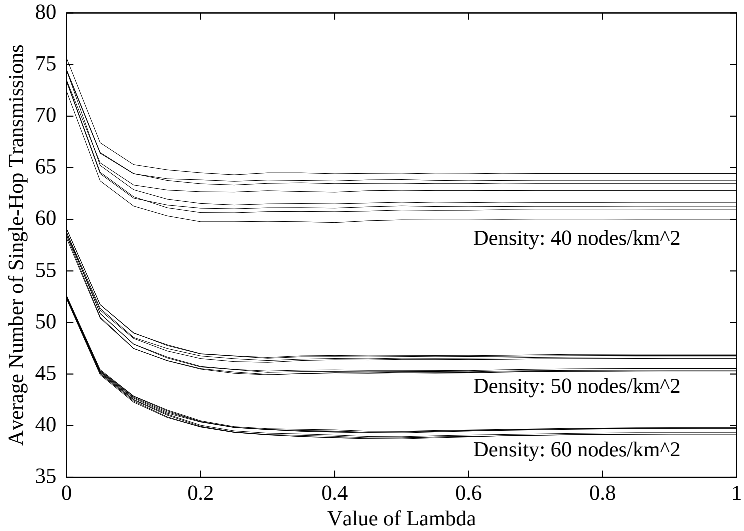


Figure 3.10: *Effect of parameter λ on the number of single-hop transmissions*

value by the number of destinations. This results in the number of single-hop transmissions that would be required had position-based unicast been used. We compared this value to the number of single-hop transmissions in PBM, with λ selected such that the number of single-hop transmissions is minimal. This was done by dividing the multicast value by the unicast value for distinct settings of area size, number of nodes, and number of destinations. Figure 3.11 shows how the reduction in single-hop transmissions increases as the number of destinations grows. The setting from which this graph was derived is a medium-sized area with a node density of 60 nodes per square kilometer and a maximum node velocity of 30 meters per second. Other combinations of parameters yield similar results, with the reduction of single-hop transmissions reaching about 66 % for 30 destinations.

In addition to reducing the number of single-hop transmissions, the usage of multicast also prevents an overload of the network close to the sender. These hot-spots appear at the sender if the same unicast packet is transmitted once per destination. While the overall load on the network is reduced by the amount shown in Figure 3.11, the reduction in those critical areas of the network is actually much higher: It is reduced by a factor depending linearly on the number of destinations.

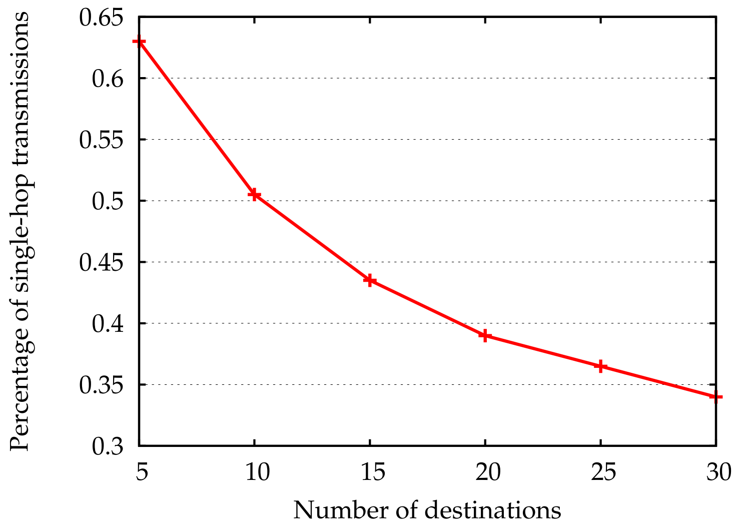


Figure 3.11: *Reduction of single-hop transmissions*

3.4 Group Membership and Position Information

In order to work as described in the previous sections, PBM requires that a forwarding node know the identity and the position of all destinations. While it is conceivable that the sender of a packet gathers this information and places it in the packet header, this does not seem to be viable for large receiver sets. In particular, this would increase the size of the header and thus limit the key benefit of multicast, i. e., the reduction of the required bandwidth. The next chapter will present a solution to this problem, introducing a location-oriented group membership management mechanism. This allows an advanced and scalable multicast routing protocol that is able to make the forwarding decision at each node without including overhead in the data packets. Furthermore, it is possible to aggregate multiple destinations that are located in one geographic region, such that the distribution of location and membership information requires only minimal resources.

This aggregation can be done hierarchically such that more detailed information about the membership and position of members becomes available as the packet approaches those members.

3.5 Conclusions

In this chapter we presented a multicast routing algorithm for mobile ad-hoc networks. Position-Based Multicast (PBM) is a generalization of existing unicast routing algorithms (e. g., face-2 or GPSR) which use the geographic position of the participating nodes to forward packets. PBM consists of a greedy forwarding part that selects the next hop(s) of a packet based on the positions of the forwarding node, its neighbors, and the destinations. Furthermore, a recovery strategy is specified for situations where greedy forwarding fails. The key advantage of PBM is that no permanent distribution structure like a tree or mesh needs to be constructed and maintained. Thus PBM is very well suited for highly dynamic networks without resorting to flooding of the data packets. The rule for splitting a multicast path includes a parameter λ that may be used to adapt the algorithm to different application scenarios by controlling the trade-off between latency and bandwidth. The application of PBM to a large number of different network parameters (node speed, network size, node density) has been investigated by means of simulation. As a consequence, the theoretical behavior of PBM in terms of drop rate, potential for bandwidth reduction, and the effect of the parameter λ is very well understood. The key issue remaining open for now is the scalable distribution of group membership and position information. The next chapter will present a solution to this problem, thereby reducing the computational complexity of the forwarding process by means of aggregation.

CHAPTER 4

Scalable Position-Based Multicast

In this chapter we present Scalable Position-Based Multicast (SPBM) [184, 185, 186]. SPBM utilizes the geographic position of nodes to provide a highly scalable group membership scheme and to forward data packets in a manner robust to changes in the topology of the network. SPBM bases the forwarding decision on whether or not there are group members located in a given direction. It allows a hierarchical aggregation of membership information. The farther away a region is from an intermediate node, the higher the level of aggregation for this region will be. Because of aggregation, the overhead for management of group membership scales logarithmically with the number of nodes and is independent of the number of multicast senders for a given multicast group. Furthermore, we show that group management overhead is bounded by a constant if the frequency of membership updates is scaled down with the aggregation level. This scaling of the update frequency is reasonable since the higher the level of aggregation is, the lower the number of membership changes for the aggregate will be. The performance of SPBM is investigated by means of simulation, including a comparison with ODMRP, and through mathematical analysis.

4.1 Introduction

Scalable Position-Based Multicast (SPBM) is an ad-hoc multicast routing protocol that comprises a multicast forwarding strategy and a group management scheme to determine where members of a multicast group are located. The forwarding strategy uses information about the geographic positions of group members to take forwarding decisions. In contrast to existing approaches, it neither requires the maintenance of a distribution structure (i. e., a tree or a mesh) nor resorts to flooding. The group management scheme uses knowledge about geographic positions to hierarchically aggregate membership information.

The forwarding of packets by means of SPBM is similar to that introduced by PBM in Chapter 3, and as such, a generalization of position-based unicast routing as proposed, e. g., in [36] and [106]. In these protocols, a forwarding node selects one of its neighbors as a next hop in a greedy fashion, such that the packet progresses towards the geographic position of the destination. The big difference, however, is that SPBM uses group addresses to deliver packets to a subset of the nodes in the network, while PBM uses all the addresses of each single destination node to distribute the data packets. This feature makes the SPBM protocol scalable in terms of the group size.

It is possible that a node will have no neighbor with progress towards the destination, although a valid route to the destination exists. The packet is then said to have reached a local optimum. In this case, a *recovery strategy* is used to escape the local optimum and to find a path towards the destination. In order to extend position-based unicast routing to multicast, SPBM provides an algorithm for duplicating multicast packets at intermediate nodes if destinations for that packet are no longer located in the same direction. This algorithm includes both greedy forwarding and the recovery strategy.

The second important element of SPBM is its group management. It relies on geographic information to achieve scalability: Instead of maintaining a fixed distribution structure, an intermediate node just needs to know whether or not group members are located in a given direction. This allows a hierarchical aggregation of membership information: The farther away a region is from an intermediate node, the higher the level of aggregation for this region can be. Thus, group membership management can be provided; its overhead scales logarithmically with the number of nodes and that is independent of the number of multicast senders in a multicast group. A second observation is then used to further reduce this overhead: The higher the level of aggregation is, the lower the frequency of membership changes for the aggregate will be. In SPBM, we therefore propose to scale down the frequency of membership update messages exponentially with the level of aggregation. This results in a constant upper bound on the overhead as the size of the network increases.

4.2 The Scalable Position-Based Multicast Protocol

We now introduce the two building blocks of our algorithm. The *group management scheme* is responsible for the dissemination of the membership information for multicast groups, to inform forwarding nodes about the directions in which receivers are located. The *multicast forwarding algorithm* is executed by a forwarding node to determine which neighbors should receive a copy of a given multicast packet. This decision is based on the information provided by the group management scheme. In the following, we assume that each node in the network is able to determine its own position, e. g., through the use of GPS.

4.2.1 Location-Oriented Group Management

Position-based multicast requires that the forwarding nodes know the locations of the destinations. Including all of the destinations explicitly in the data packet header does not scale well as the size of the multicast group increases. To improve scalability, our proposal introduces hierarchical management of group memberships.

To this end, the network is subdivided into a quad-tree with a predefined maximum level of aggregation L. Figure 4.1 shows a quad-tree with four levels. Single squares are identified by a number that is composed of their square numbers on all levels from n to 1. In the example, the identifier "442" identifies a level-0 square that is located in the level-3 square comprising the whole network, in the level-2 square "4" and in the level-1 square "44". In level-0 squares, by definition, all nodes are within radio range of each other (i. e., level-0 squares have at most a diameter the size of the radio range).[1]

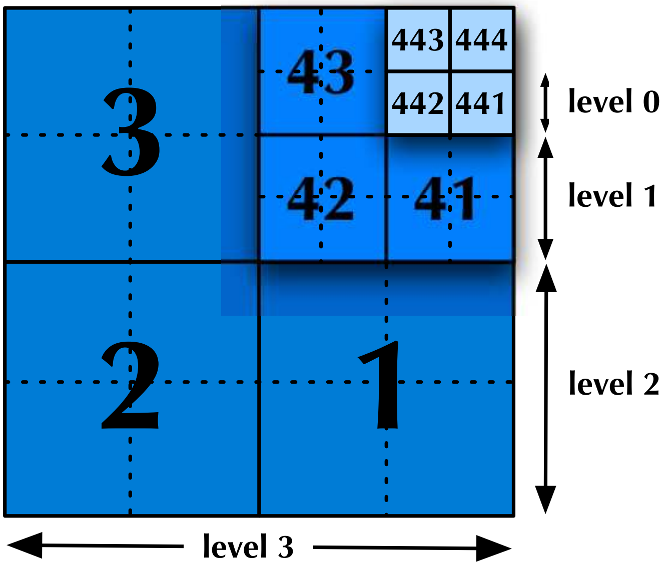


Figure 4.1: *Network represented by a quad-tree* ($L = 3$)

Algorithm

The membership update mechanism aims to provide each node in the ad-hoc network with an aggregated view of the position of group members. Each node maintains a global member table containing entries for the three

[1]For the design of the protocol, we assume the Unit Disk Graph radio model. There is no model that perfectly describes reality, and thus it is reasonable to choose this simple model, which also allows us to provide a strong theoretical analysis.

neighboring squares for each level from level 0 up to level $(L-1)$. In addition, each node has a local member table for nodes located in the same level-0 square.

Each entry in the global member table consists of the square's identifier and the aggregated membership information for all nodes in that square. Each entry in the local membership table consists of a node ID and the membership information for that node. Additionally, every entry in the tables is assigned a validity time. Membership information is stored and transmitted as membership bit vectors. For the sake of simplicity, we assume that each bit represents one multicast group. A bit that is set to 1 indicates group membership. This encoding enables us to accommodate 256 groups in 32 bytes. If there is a need for a much greater number of groups, a Bloom filter [33] scheme could be applied, in which the number of encodable groups is greater than one per bit, trading in the possibility of false positives, i. e., a multicast group could be falsely believed to have receivers in a square.

With the simple bit vector scheme, the amount of state maintained in a node scales logarithmically with the size of the network. Table 4.1 shows an example of a node located in square "442" with a membership vector length of 8 (for groups 0 to 7). In this example, the first entry in the global member table can be interpreted as follows: There is at least one multicast receiver for groups 3, 4 and 5 located in the level-2 square "1". The first entry in the local member table contains the information that node 14 is in the same level-0 square as the node maintaining the table, and that 14 is member of group 7. Validity times are omitted here since they do not affect the following algorithmic details.

A node indicates its group membership status by broadcasting *announce* messages within its level-0 square (i. e., to its direct neighbors). An announce message contains the ID of the node, its position, and a membership vector describing its subscribed groups. Announce messages are broadcast periodically, but do not need to be forwarded by any other node since all nodes within the same level-0 square are within radio range of each other. These messages replace the beacon messages of position-based routing.

Table 4.1: *Global and local member table of a node located in square "442"*

Square	Groups
1	00011100
2	01000000
3	10100010
41	01010100
42	00010001
43	00100000
441	00000000
443	00010100
444	00100100

Node	Groups
14	00000001
23	01000100
51	00000100

A node stores the membership information for all nodes in its level-0 square. Update messages are then used to provide all nodes that are located in a level-1 square with the aggregated membership information for the four level-0 squares contained in the level-1 square. This is done by periodically selecting one node in each level-0 square. For now, we assume that such a selection mechanism is in place. We will show later how it can be realized by means of random timers. The selected node floods the level-1 square with an update message that includes the ID of the selected node, a membership vector describing the aggregated group membership information, the identifier of the destination square that is to be flooded, and a sequence number so as to enable duplicate message detection. The aggregation is done by a bitwise disjunction ('OR') of the membership vectors of the nodes located in the level-0 square. In order to perform flooding, each node in the level-1 square forwards this message once. Thus, a total of four update messages will be flooded in each level-1 square per period: one from each level-0 square. In the example, one node in each square "441", "442", "443", and "444" is selected. Those nodes aggregate their level-0 membership information and flood them in an update packet within the level-1 square "44".

The same mechanism is used to aggregate the membership information from an arbitrary level-λ square and flood it in the area of a level-$(\lambda + 1)$

square. In the example, one node in each square "41", "42", "43", and "44" was selected to aggregate its level-1 membership information and flood an update message within square "4". If the node with the membership tables depicted in Table 4.1 were to be selected for square "44", it would perform the aggregation by a bitwise OR operation on the membership vectors for the individual nodes 14, 23, and 51, and on the aggregated information from the level-0 squares "441", "443", and "444".

Since the size of a square increases exponentially with each level, the likelihood that the aggregated group membership information will change in a given time-span decreases rapidly. We therefore propose to decrease the frequency of membership information flooding exponentially with the level of aggregation. Let f_0 be the frequency of announce messages. Then the frequency f_λ of update messages from a single square on level λ is defined as follows:

$$f_\lambda = q^\lambda \cdot f_0 \qquad \text{for} \qquad \lambda = 1,\ \ldots,\ L \quad \text{and} \quad 0 < q \leq 1,$$

where q is a selectable factor that determines the decrease of the update frequency from one level to the next. All update messages have a validity time 2.5 times the length of the update interval of the respective level. After this validity time has expired, the corresponding entry is eliminated from the membership table. This avoids the use of outdated information should no current messages have been received.

It remains to be shown how one node is selected to send an update message. The selection mechanism is performed by random timers. Every node maintains an update timer for each level. When the timer expires, the node is selected, the update message for the appropriate level is transmitted, and the timer is reset. When a node receives an update message for a square to which it belongs, its timer is reset without sending the packet, thereby suppressing the transmission of the update message. The main component of each timer is determined by the update frequency of that level. In order to prevent all nodes in a given square from flooding the same update information simultaneously, each timer also has a random exponential element. It was shown in [143] that exponentially distributed timers lead to good suppression ratios, even for large groups of nodes. To generate an exponential distribution, we use the inverse transform of the corresponding distribution function and feed it with random values x that are uniformly distributed between 0 and 1:

$$f(x) = -\frac{T}{\beta} \cdot \ln\left(x \cdot \left(e^{\beta} - 1\right) + 1\right) \quad \text{for} \quad 0 \leq x \leq 1,$$

where T determines the interval from which the timers are chosen, and β is a parameter that, while increasing, shifts the weight of the resulting distribution towards the end of the interval. The total runtime of a timer for a given level λ is then chosen as follows:

$$t(x) = \left(\frac{1}{f_0} + \frac{T}{\beta} \cdot \ln\left(x \cdot \left(e^{\beta} - 1\right) + 1\right) - E[M]\right) \cdot \left(\frac{1}{q}\right)^{\lambda} \quad \text{for} \quad 0 \leq x \leq 1.$$

The first part of this equation, $\frac{1}{f_0}$, describes the basic update interval at level 0. Added is an exponentially distributed random value between 0 and T. The parameters T and β can be used to tune the probability of collisions: Lower values for both of them lead to tightly chosen timers and thus likely result in collisions, whereas higher values increase the expected value and thus create additional delay. [143] suggests a β value between 0 and 10, T has been set to half of the basic update interval which is $\frac{1}{2f_0}$. Having chosen the parameters, the expected value of the minimal timer can be estimated as, according to [143],

$$E[M] = T \cdot \int_0^1 \left(1 - \frac{e^{\beta m} - 1}{e^{\beta} - 1}\right)^R dm,$$

where R is the number of competing nodes. The resulting value is then scaled by the factor $\frac{1}{q}$, powered by the level λ. Thus, the exponential part also depends on the level, and therefore on the area in which the nodes should be suppressed. Through the exponential distribution, the probability of having a low timeout value is much less than the probability of a high timeout value. Thus, the vast majority of timers will not expire before an update message from another node has been received. Note that the largest part of the timer is deterministic. The random component used for the selection process therefore has no significant impact on the frequency with which squares are flooded. Given a constant node density, it can be shown that the amount of data transmitted per square meter by this group management scheme is bounded by a small constant, as the size of the network increases to infinity.

Scalability Analysis

The group management algorithm is proactive; thus, its overhead is independent of actual data traffic and the number of senders in a given multicast group. In the following, we quantify this overhead to examine the algorithm's performance and scaling characteristics.

Let the radio range r be constant. To ensure connectivity within level-0 squares (under the assumption of a unit disk graph), the area A_0 of level-0 squares (see Figure 4.2) is:

$$A_0 \leq \frac{r^2}{2}$$

and the area covered by the network with respect to the number of levels can be determined as:

$$A(L) = A_0 \cdot 4^L.$$

We need to determine how often a level-0 square is flooded with update messages from all levels in a fixed period. In a first step, let us consider the case that $q = 1$ and the update frequency is therefore the same for all levels. Then, at level 0, four update messages are generated by four squares which form a level-1 square. These messages are received by each node within the level-1 square. The same holds for each level from 1 up to $L - 1$. Thus, the overhead c linearly depends on the number of levels L. If we quadruple the area of the network, thereby increasing the number of levels by one, each single lowest-level square has to be flooded with four more messages. This

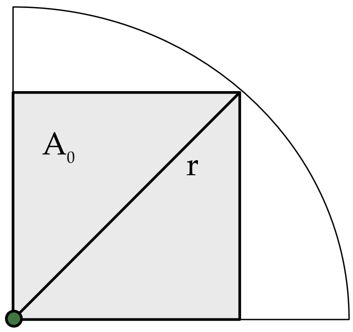


Figure 4.2: *The area of a level-*0 *square*

means that a multiplication of the size of the network area A stresses only a single node with a constant additional load.

Considering the spatial frequency reuse occurring in a network of growing size, we study the overhead per area. In terms of complexity, the total cost c[2] per area in the network conforms to

$$\frac{c(A(L))}{A(L)} = O\left(\log A(L)\right),$$

which means that the cost per area only grows logarithmically with the size of the area.

More generally, if $0 \leq q \leq 1$, and if n is the number of nodes in the network, then the total cost of update messages in the network c is

$$c(L) = n \cdot f_0 \left(1 + 4 \sum_{\lambda=1}^{L} q^{\lambda}\right),$$

where again f_0 is the frequency of the update messages at level 0 and q the factor which decreases this frequency from one level to the next.

Theorem 1 (Cost Function)**.** *Consider an ad-hoc network of square geometry containing n nodes. Let L be the maximum hierarchy level, $0 < q \leq 1$ the timer frequency coefficient, and f_0 the smallest-square frequency. Then the average num-*

[2]This cost metrics assumes constant-length update messages. If we increase the number of multicast groups, the length of the messages grows linearly.

ber of (proactive) radio transmissions per time of the SPBM group management protocol is given as

$$\begin{aligned} c &= nf_0\left(1+4\sum_{\lambda=1}^{L} q^{\lambda}\right) \qquad (4.1)\\ &= \begin{cases} nf_0\,(1+4L) & q=1 \\ nf_0\left(1+4\left(\frac{1-q^L}{1-q}\right)\right) & 0<q<1. \end{cases} \end{aligned}$$

Proof. Let c_λ be the average number of transmissions per second on level λ ($\lambda = 0, \ldots, L$) for the whole network. At level 0, each node sends f_0 packets per second. Thus

$$c_0 = nf_0.$$

At every higher level λ ($\lambda = 1, \ldots, L$) $4^{L-\lambda}$ squares exist, each with $\frac{n}{4^{L-\lambda}}$ nodes on average. At a frequency of f_λ, one of the nodes of each square at level λ sends update packets. Each of these packets is relayed by all nodes in the four adjacent squares of level λ which belong to the same square of level $\lambda+1$. This leads to $\left(4\frac{n}{4^{L-\lambda}}\right)$ packet transmissions for each square of level λ:

$$c_\lambda = 4^{L-\lambda}\cdot 4\cdot \frac{n}{4^{L-\lambda}}\cdot f_\lambda = 4\cdot n\cdot f_\lambda \qquad (\lambda = 1, \ldots, L).$$

Aggregating the cost on all levels, we have

$$\begin{aligned} c &= c_0 + \sum_{\lambda=1}^{L} c_\lambda \\ &= nf_0 + \sum_{\lambda=1}^{L} 4nf_\lambda \\ &= n\left(f_0 + 4\sum_{\lambda=1}^{L} f_\lambda\right). \end{aligned}$$

Incorporating the definition of the frequencies $f_\lambda = q^\lambda f_0$ gives

$$c = nf_0 \left(1 + 4\sum_{\lambda=1}^{L} q^\lambda\right),$$

leading directly to the theorem. □

If $0 < q < 1$, the sum in Equation 4.1 represents a geometric series which has an upper limit for all values of L. Thus, for $q < 1$, the total cost per area in the network is bounded by a small constant number of update messages per time when growing the area of the network:

$$\frac{c(A(L))}{A(L)} = O(1) \qquad \text{if} \quad q < 1.$$

Corollary 1 (Cost Complexity)**.** *With the definitions of Theorem 1, the SPBM group management protocol overhead per area and time has a complexity of $O(1)$ for $q < 1$ and $O(\log A)$ for $q = 1$ with respect to the total size of the network area A.*

Proof. Let us assume that we have a network consisting of only one square of size A_0. We further assume that we have a limited node density d denoting the number of nodes per A_0 area. The number of nodes in the complete network is then given as $n = dA$, where A is a multiple of A_0. Whenever the network area increases, we quadruple the network area by increasing the hierarchy level by one, such that the new area is covered by the new square, i. e., the number of hierarchical levels is calculated as

$$L(A) = \lceil \log_4 A \rceil < 1 + A_0 \log_4 A. \tag{4.2}$$

With the increase of the area, the possibility of spatial frequency reuse grows linearly. Thus, we consider the cost per area c_A. Following Equation 4.1, the average overhead cost per time and area is

$$\begin{aligned} c_A &= \frac{c}{A} \\ &= df_0 \left(1 + 4\sum_{\lambda=1}^{L} q^\lambda\right). \end{aligned}$$

Using Equation 4.2, an upper bound $\overline{c_A}$ for the cost per area can be specified:

$$\overline{c_A} = df_0 \left(1 + 4 \sum_{\lambda=1}^{1+\log_4 A} q^\lambda\right).$$

Considering the case $q = 1$, this upper bound results in

$$\overline{c_A} = df_0 \left(5 + 4\log_4 A\right),$$

which conforms to $O(\log A)$.

Since $0 < q < 1$, the geometric row converges and is bounded:

$$\begin{aligned} c_A &= df_0 \left(1 + 4\sum_{\lambda=1}^{L} q^\lambda\right) \\ &< df_0 \left(1 + 4\frac{q}{1-q}\right), \end{aligned}$$

which is independent of the chosen area size or the maximum level, respectively. Thus, the complexity is $O(1)$. $\square$

On the other hand, the worst-case join latency grows exponentially with the number of levels, or linearly with the network size. This worst case occurs if a node joins a group where a sender resides in a different level-L square and the following situation arises: The node has just sent an update message at level 0 and, for each other level λ, the update at level $\lambda + 1$ was sent just before the update at level λ arrives. Thus, at each level a full update period elapses before the membership is propagated the next level up. The sum of all timer durations counts towards the latency until the join is completed:

$$t_{\text{timer}}(L) \leq \sum_{\lambda=0}^{L} \frac{1}{q^{\lambda} \cdot f_0} = \frac{1}{f_0} \cdot \frac{1 - \left(\frac{1}{q}\right)^{L}}{1 - \frac{1}{q}}.$$

This scales exponentially with the number of levels L and, since

$$L = \log_4 \frac{A(L)}{A_0},$$

linearly with the size $A(L)$ of the network:

$$t_{\text{timer}}(L) = O\left(e^{L}\right) = O\left(A(L)\right).$$

Additionally, the worst case when selecting the propagating node would be that the selected node is located in the corner of the square that is farthest from all other points of the area that has to be flooded. The maximum distance scales with the diameter of the network, and the total time for the distribution of an update message thus conforms to

$$t_{\text{tx}}(L) = t_{tx}^{0} \cdot \sum_{\lambda=0}^{L} 2^{L} = t_{tx}^{0} \cdot \left(2^{L+1} - 1\right) = O(e^{L}),$$

which does not further influence the complexity of the worst case join latency:

$$t_{\text{join}}(L) = O\left(A(L)\right).$$

This worst-case join latency has to be accepted in favor of the good scaling properties of the network load for larger networks. The average join delay, however, is much shorter since the worst case occurs only if a joining receiver is the first receiver of a certain group in a highest-level square. The membership information has to be propagated only up to the level where other receivers have already subscribed. For dynamic groups this means that high latencies do not occur if there is always a certain number of receivers distributed in the area where the group communication takes place.

The same holds for mobile receivers. A receiver which enters a square without any subscribed nodes has to wait until its membership has been propagated at least to the level where the senders or further receivers of this group are. However, if a receiver crosses a square boundary to a square where other nodes are already subscribed, it will be able to receive multicast messages immediately after sending its first announce beacon.

Alternative: Reactive Group Management

The key advantage of the described group management is that its overhead is bounded. However, there are two possible drawbacks: (a) A high worst-case join latency may occur and (b) even if no multicast sessions take place, group management messages are transmitted and cause load in the network. We thus propose an alternative group management [39] which provides the same membership information to the participating nodes but works in a reactive way. It propagates membership updates whenever a node joins a group or crosses a square boundary. Group leaves are handled in a soft-state manner; group joins have only a limited validity and have to be refreshed on a regular basis. Overall, the approach described in this paragraph yields greater accuracy in membership information than the proactive algorithm.

The reactive algorithm uses two separate membership tables. The first one will store the updates sent to the network (*sent table*). The second one stores the received membership updates (*received table*). Both tables store update atoms which consist of a group, a square that is subscribed to this group, and a validity time. When the validity time has elapsed, the entry may be removed. An update packet consists of one or several such update atoms. Whenever an update is received, it will be added to the received updates table. Instead of simply redistributing the update message, the receiving node checks whether there is a difference between the membership information stored in the sent table and the membership information stored in the received table. Such a difference may be either a group join for a square or a longer validity for a square. Relevant are only entries for squares the node has to be aware of (or aggregates of the entries for sub-squares of these squares), and only membership information for these squares will be sent out. If there are differences between the tables the differing entries will be sorted in the order of their importance: most important are those entries that are not contained in the sent table at all, followed by entries with a small residual validity—they need to be refreshed more urgently. Beginning with the most important updates, an update packet is created containing all updates or as many updates as fit into the packet.

It remains to describe how the validity values are calculated. On level 0, i. e., in the smallest squares, movement prediction is used. A node joining a group chooses a validity according to the expected point in time at which it will leave the square—at most a predefined maximum validity. The validity times of memberships for squares on higher levels are calculated upon sending an update for such a square. This is done based on all existing entries for its sub-squares. If an update in the received table is an update for a sub-square, the corresponding validity is multiplied by a factor s^m, where $s > 1$ and m is the difference between the levels of the square itself and the level of the sub-square the update comes from. The resulting validity for the high-level square is then determined as the maximum of the validity values of all sub-squares that belong to that square.

Let us look at an example. A node that wants to send an update for square "1" previously received an update for square "131". It calculates the validity of the memberships for square "1" as s^2 times the validity of square "131"'s entry because square "131" is 2 levels lower than square "1".

The membership management variant chosen depends on the expected join behavior and movement of the nodes in the network. Few membership changes and a low mobility favor reactive management, while proactive management pays respect to a high agility that would otherwise cause a high amount of generated update messages.

4.2.2 Multicast Forwarding

In the following we will assume that the proactive membership management is in place; equivalent information may be provided by the reactive algorithm described in the previous paragraphs.

To deliver multicast packets from a source to the subscribed members of a group, the nodes use the information stored in their member tables. By dividing the network into a quad-tree, geographic regions are built which can be used to aggregate multicast traffic to group members located geographically close to each other.

The forwarding decision is based on information about neighboring nodes. Each node maintains a table of the nodes in its transmission range. This is accomplished by having each node periodically broadcast beacon messages containing their ID and their position. Beacon messages are not forwarded by the receiving nodes.

Algorithm 1 shows how forwarding works. As input, the algorithm requires the current node n, the packet p, and the list of neighbors N of n. The packet includes a list of destinations, which initially contains one entry comprising the whole network, and a group address indicating the group to which the packet is being sent. When the algorithm is invoked, it first checks whether the current node n is a member of the multicast group the packet is being sent to. If so, the packet is delivered.

In the next step, the algorithm looks at each entry in the list of destinations in the packet: If the global or the local membership tables contain a de-aggregation of the entry, then the entry is subdivided into those squares of the next lower level that include members for the group to which the packet is being transmitted. At level 0, a de-aggregation is performed by replacing the square with the IDs of the nodes that are group members.

Consider, for example, the situation where a node in square "442" (see Figure 4.1) sends a multicast packet to the group number 1. It initializes the packet with the whole network as the single destination area and sets the multicast address to 1. The packet is then handed to the forwarding algorithm. After checking whether the current node is a receiver of multicast

Algorithm 1 The forwarding algorithm

Require: node n, packet p, list of neighbors N

  if $n \in receivers(group(p))$ **then**
    $deliver(p)$
  end if
  $D \leftarrow \emptyset$
  for all $d \in destinations(p)$ **do** {for all destinations packet p is addressed to}
    if $mysquare \subseteq d$ **then** {$mysquare$ is the current level-0 square of node n}
      $D \leftarrow D \cup subdivide(d)$ {replace destination d by its sub-squares}
    else
      $D \leftarrow D \cup \{d\}$
    end if
  end for
  $F[N] \leftarrow \emptyset$ {stores the destinations for a given next hop}
  for all $d \in D$ **do**
    $\nu \leftarrow \emptyset$ {stores the next hop}
    if $recover(d)$ **then** {is destination d in recovery mode?}
      $\nu \leftarrow rightHand(prevHop,\ d)$
    else
      $\nu \leftarrow forwardGreedy(N,\ d)$
    end if
    if $\nu = \emptyset$ **then**
      $\nu \leftarrow rightHand(n,\ d)$
      if $\nu = \emptyset$ **then**
        $drop(d)$
      end if
    end if
    $F[\nu] \leftarrow F[\nu] \cup \{d\}$
  end for
  for all $\nu \in N$ **do**
    if $F[\nu] \neq \emptyset$ **then**
      $send(p,\ \nu,\ F[\nu])$ {send packet p to neighbor ν for destinations $F[\nu]$}
    end if
  end for

group 1, the destinations are de-aggregated. Based on the membership tables given in Table 4.1 for multicast group 1, the complete network can be de-aggregated into the level-2 square "2" (since bit 1 of the membership vector is set), the level-1 square "41", and the individual node 23 in the same level-0 square as the forwarding node.

After de-aggregation of the destinations, it is checked which neighbor is best suited to forward the packet to each destination. This is done in a fashion similar to position-based unicast routing (see [135]): In order to determine the most suitable next hop for a packet and a given destination, the source compares the geographic progress of each of the neighbors with respect to the destination and picks the neighbor with the best progress. If the destination is a square, the position of the nearest point in that square is used as the destination position.

After finding the next hop for each destination, the current node n makes a copy of the data packet for each of these next hops. In the list of destinations, it enters a list of the destinations which shall be reached through this specific next hop, and sends the packet to the next hop by means of unicast transmission. The use of unicast increases the reliability of data delivery at the expense of bandwidth utilization, as each copy of the packet will be acknowledged on the MAC layer, but at the cost of multiple messages. In Chapter 6 we will investigate the use of broadcast for single-hop forwarding.

Figure 4.3 shows an example of the forwarding procedure.[3] Node A wants to send a packet to the group of which nodes C, E and F are members. Thus A's member table contains the information that there is at least one receiver in square "4". It sends the packet in this direction, and node B is the first node located in the level-2 square "4". Consequently, it has the information that there are nodes subscribed to the group in the level-1 squares "41" and "44". It therefore updates the information in the packet header accordingly. Node C is the first forwarding node in square "41". Besides delivering the packet, it checks its member table and recognizes that it does

[3]The figure depicts only nodes which are involved in the process of refining the destination square information.

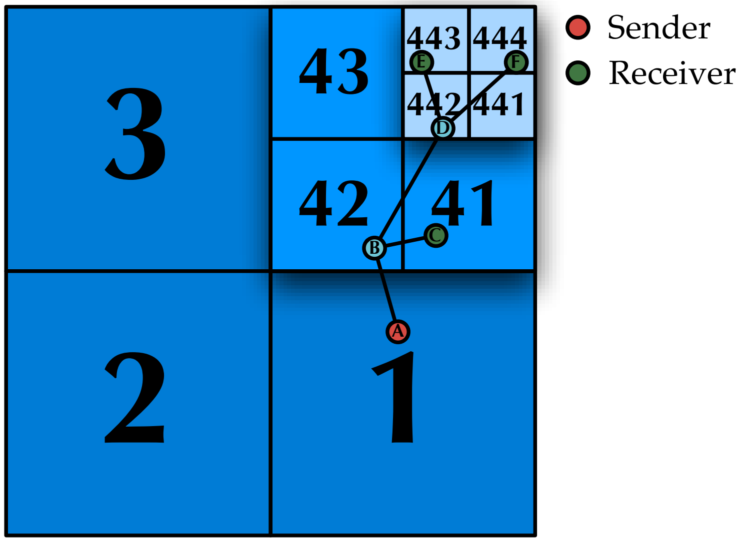


Figure 4.3: *Forwarding in the quad-tree*

not need to forward the packet to any additional receivers in square "43". In square "44", node D replaces square "44" in the packet header with the level-0 squares "443" and "444". After receiving the packet, nodes E and F replace their square with potential additional destination nodes in this square. If there are any, the packets will now be sent directly to the receivers since the radio ranges of E and F cover the complete squares "443" and "444", respectively.

Recovery from Greedy Failures

If, for one or more destinations, a forwarding node does not find a next hop that yields geographic progress, a recovery strategy has to be employed. Similar to position-based unicast routing [106, 36], SPBM uses a distributed planarization of the network graph combined with the right-hand rule to route around void regions. If there is a destination with no suitable next hop, the algorithm first planarizes the surrounding network graph. Then, the node determines the angles counter-clockwise between the line from the node to the destination, and the line from the node to each remaining neighbor. The neighbor with the smallest angle is chosen as the next hop. This destination is marked as a *recovery destination*, and the current position is stored in the packet in order to inform the following hops about the position where the recovery mechanism started. The chosen next hop is then handled as a normal destination.

Upon receipt of a packet containing a recovery destination a node first checks whether it is itself located closer to the destination than the position which is stored in the packet as the recovery starting point. The destination is always known by every node in the network since the recovery mode is only needed for destination *squares*, whose positions are known by definition. In this case, regular forwarding is resumed. If this is not the case and the node is located farther away from the destination than the recovery starting point, the node has to continue the recovery process. After performing planarization, it chooses the next hop according to the right-hand rule.

The recovery strategy works independently of the grid structure. As long as a destination is marked as a recovery destination, it is not necessary to change or replace it because only the nodes at the destination have enough information to refine the destination square.

Figure 4.4 shows an example of the case in which one destination cannot be reached using a greedy strategy. In this example, a packet arriving at node C is addressed to the three destination squares I, II, and III, which are depicted as the shaded areas. Direct communication is only possible where the nodes are connected by a line. Thus, node C selects B as a forwarder to destination III, and D as a forwarder to destination I. Since there is no node with geographical progress towards destination II, this destination has to be handled as a recovery destination. Applying the right-hand rule, node D is selected as the next hop to destination II. In the following, C sends a packet to node D addressed to the destinations I and II where II is marked as a recovery destination. Destination III is reached in a greedy fashion and shall no longer be part of this example. Node D now checks whether it is closer to the recovery destination II than the node which put the destination into this mode. Since it is not, destination II remains in recovery mode. The

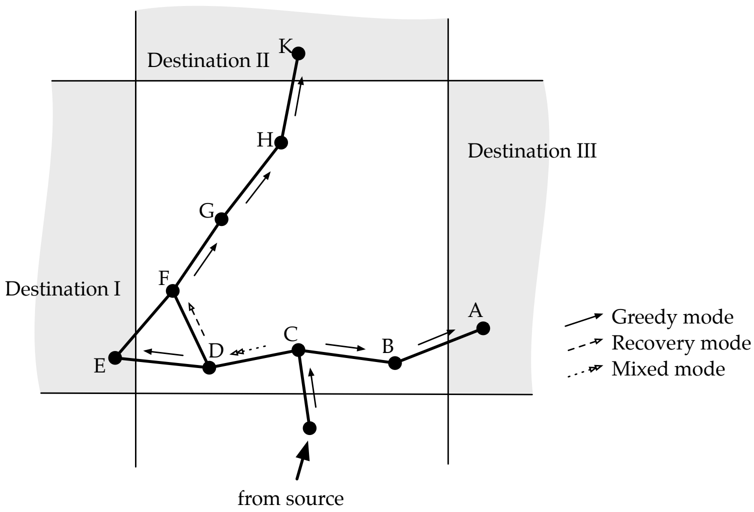


Figure 4.4: *Recovery from a greedy failure*

next hop according to the right-hand rule is node F, whereas destination I can be reached greedily via node E. The packet from D to F contains only the recovery destination II. Arriving at node F, the packet is closer to the destination than at the point where the recovery mode was started. Hence, it can be switched back into greedy mode and routed to K via H.

4.3 Evaluation

4.3.1 Simulation Setup

The simulations were performed using the network simulator ns-2 [5].[4] The ODMRP implementation from [6] was chosen as a reference and ported to ns-2.27. We discovered and rectified some misbehaviors of this implementation: First, the calculation of the header sizes was corrected; second, the delay for join queries was changed so as to contain a constant part in addition to the random backoff. This reduces the overhead by about 10 %

[4] In contrast to PBM (see Section 3.3), the forwarding decision in the SPBM protocol has a much lower computational complexity. Thus, it is feasible to perform simulations with ns-2 allowing packet level simulation and investigation of the protocol's operation.

without affecting the packet delivery ratio. Third, according to the ODMRP draft [123], duplicate join queries and join replies are suppressed, further reducing the overhead of the protocol.

The MAC layer in all simulations was IEEE 802.11, with a maximum bandwidth of 2 MBit/s. The transmission power resulted in a radio range of 250 meters. Since the transmitted packets were relatively small, the use of RTS/CTS was disabled. All runs were simulated 20 times with different random seed values and in different movement scenarios; we report on the average of those runs, the vertical bars in the graphs showing the variance of the results. A run represents the simulated time of 180 seconds, where nodes joined at the beginning of the simulation, and the first data packet was sent after 60 seconds in order to give the group management enough time to initialize. The data payload size was 64 bytes per packet, and each source transmitted one packet per second.

The protocol-specific parameters of SPBM were as follows: The basic group membership update frequency for an level-0 square f_0 was set to $\frac{1}{3\,s}$. The value for the timeout of entries in the member table was 2.5 times the corresponding update interval. ODMRP's protocol-specific parameters were: a 3-second join refresh interval, a 25-millisecond acknowledgment timeout for join table messages, and a maximum of 3 join table transmissions. To improve comparability, all these protocol-specific parameters were kept constant throughout all simulations.

Some other simulation parameters were varied to investigate their influence on the results. During each series of simulation runs, only one parameter was changed. The varied parameters were: The modeled scenarios were squares measuring from 350 m × 350 m to 2,800 m × 2,800 m, where 100 nodes per square kilometer moved according to the Random Waypoint (RWP) mobility model as described by [101, 38], with a pause time of 10 seconds and, for mobile scenarios, a minimum speed of 1 meter per second. The maximum speed varied from 0 to 15 meters per second.[5] The number

[5] Again, the effects described in [26, 40, 197, 27] (mobility decrease and non-uniform node distribution after some minutes) did not occur in the simulations described here since the scenarios only had a duration of 180 seconds.

of senders ranged from 1 to 15, and the number of receivers from 5 to 25. While all senders and receivers belonged to one multicast group, they were disjoint.

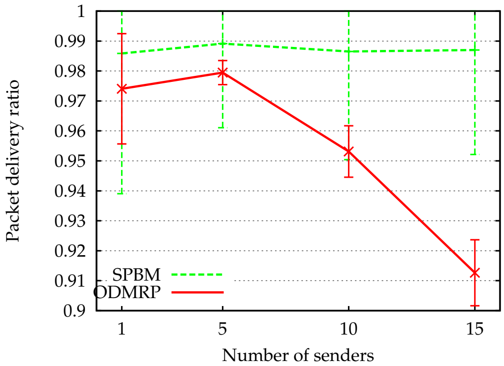
4.3.2 Performance Metrics

The metrics used to evaluate the protocol performance are packet delivery ratio, overhead, and delay. The *packet delivery ratio* (PDR) is defined as the sum of all unique data packets received, divided by the sum of all data packets that should have been delivered (sum of sent packets multiplied by the number of receivers).
The *overhead* is the total number of bytes transmitted at the MAC layer, including acknowledgments in the case of unicast transmissions. To measure the overhead on the MAC layer, it is necessary to capture MAC layer retries induced by mobility or packet collisions. These effects would be invisible if the overhead were counted at the network layer.
The *delay* is defined as the interval that elapses between the time a packet is sent and the time at which the packet is successfully delivered. This value is averaged over all packets and all receivers.

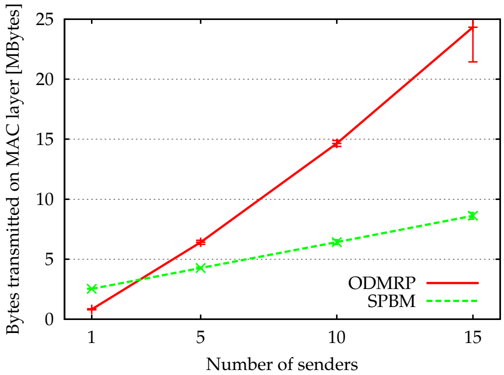
4.3.3 Results

Number of Senders

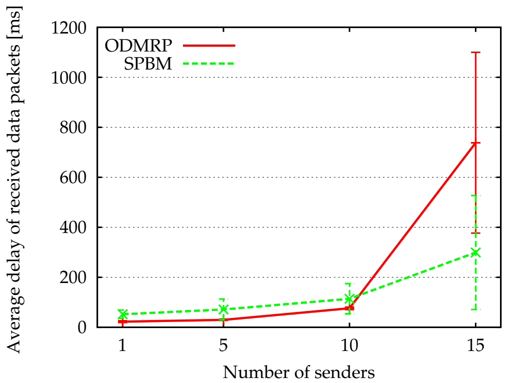
Figure 4.5 shows the respective PDR, overhead, and delay for an increasing number of senders. The other parameters were kept constant in this setup. While the PDR of SPBM is quite stable for different numbers of senders (up to 15 in these experiments), ODMRP suffers from the load generated by the additional senders. This is due to the fact that each sender floods the entire network with data and control packets at regular intervals in order to build its forwarding group. The group management of SPBM is independent of the existing multicast sources. If only one sender is active, the network load induced by ODMRP is lower than with SPBM. This is because the proactive group management of SPBM is responsible for maintaining a certain



(a) Packet delivery ratio



(b) Bytes transmitted on MAC layer



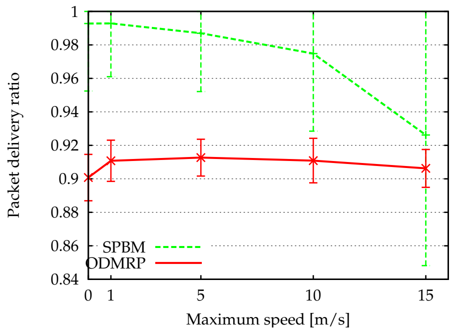
(c) Delay

Figure 4.5: *Performance with respect to number of senders (25 receivers, 1 Pkt/s, 5 m/s, 1400 m × 1400 m, 100 nodes/km²)*

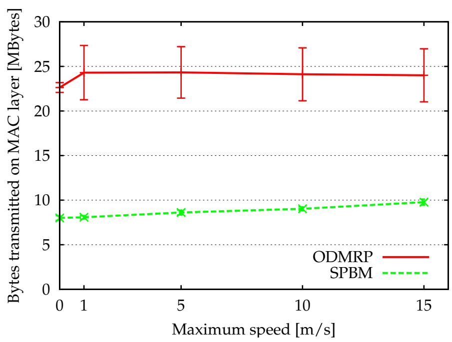
constant overhead (in these experiments the cumulated average bit rate of update messages is about 6.89 kBit/s). For ODMRP, the high increase in load is accompanied by a high decrease in the ratio of delivered packets.
SPBM, in contrast, sustains a satisfactory packet delivery ratio. The increase in overhead is mainly due to the increased number of forwarding operations for the data packets of the additional senders. The proactive group management overhead of SPBM remains constant while the number of announce messages decreases. This is due to the use of implicit beaconing, where information for announce packets is prepended to data packets whenever possible.
A similar result was achieved when varying the number of receivers while keeping the number of senders constant. ODMRP quickly saturates the network, resulting in a constantly heavy network load, while SPBM still operates at a satisfactory packet delivery ratio, with a load increase caused mainly by a higher number of forwarding operations.
Regarding the end-to-end-delay (see Figure 4.5(c)), the results show that ODMRP performs slightly better than SPBM if the number of senders is small. Since ODMRP's forwarding algorithm is a form of scoped flooding and the delay is measured as the first copy of a certain packet arrives, ODMRP is able to use the direct route from the source to each destination. At the same time, the overhead introduced through the scoped flooding leads to a steep increase in the delay once the network becomes saturated due to the increase in senders.
However, the hierarchical approach of SPBM entails routes that in some cases are slightly longer than the optimum. But once the load in the network has reached a certain level, ODMRP is no longer able to deliver packets in a timely manner. SPBM, in contrast, shows no weaknesses regarding the delay with increasing network load.

Node Mobility

Figure 4.6 shows the impact of node mobility on the packet delivery ratio and the bytes transmitted at the MAC layer. While SPBM performs very



(a) Packet delivery ratio

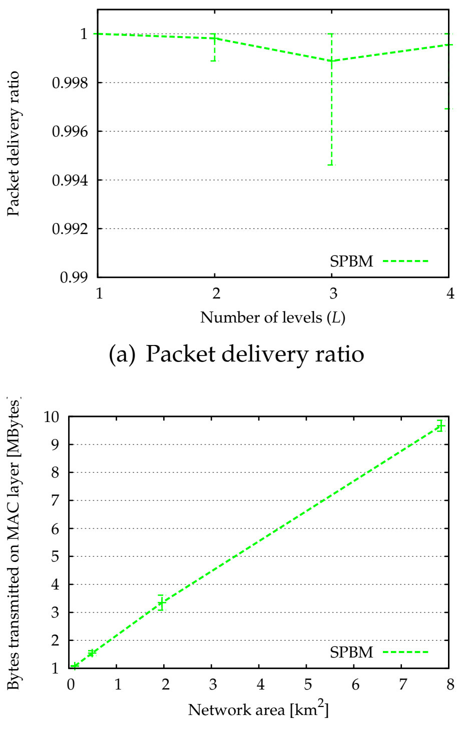


(b) Bytes transmitted on MAC layer

Figure 4.6: *Performance with respect to maximum movement speed (15 senders, 1 Pkt/s, 25 receivers, 100 nodes/km^2)*

well for low-to-medium node mobility, the packed delivery ratio drops significantly at high node speeds. Further investigation revealed two reasons for this behavior: (1) When group members cross square "boundaries" into a square that did not previously contain a group member, they will not receive packets until the group management scheme has spread the new information. (2) When node mobility increases, forwarding failures appear that are induced by discrepancies in the neighbor table used for the next-hop selection. If a node is selected as a forwarder but has moved out of radio range, the current forwarder has to wait for four unsuccessful retransmissions followed by a link layer notification before it is able to select a different node.[6] This reduces the packet delivery ratio and increases the number of MAC packets transmitted. To avoid this problem, we will adapt the ideas of contention-based forwarding, as described in [71], to SPBM. The next chapter will describe the contention-based multicast protocol CBMF that makes both the delivery rate and the number of transmitted packets largely independent of node mobility.

Network Size



(a) Packet delivery ratio

(b) Bytes transmitted on MAC layer

Figure 4.7: *Performance with respect to network size (3 senders, 1 Pkt/s, 10 receivers, 100 nodes/km^2)*

In these experiments we varied the size of the network from 350 m × 350 m to 2,800 m × 2,800 m. This corresponds to L (numbers of levels) from 1 to

[6] This effect has been extensively described in [71].

4. The node density was left constant, as was the number of senders and receivers. Thus, the number of nodes ranged from 13 for $L = 1$ to 784 for $L = 4$. Figure 4.7(a) indicates that the packet delivery ratio is independent of the number of levels (or the network area). As expected, the network load grows linearly with the area of the network (see Figure 4.7(b)). Because of the limitations of ns-2, it is not easily possible to perform simulations with larger scenarios. However, the obtained results exactly comply with the results of the theoretical analysis and it is to be expected that larger scenarios behave similarly.

4.4 Conclusions

We described in this chapter a novel position-based ad-hoc multicast routing protocol. It differs significantly from previous work in that it introduces a hierarchical organization of nodes to manage membership, as well as to forward packets. By means of simulation we demonstrated that SPBM performs very well, particularly if the number of multicast senders and receivers increases.

We are convinced that a hierarchical approach to position-based multicast is a very promising solution if the protocol is intended to scale to a reasonable number of nodes.

The biggest problem of the protocol presented in this chapter is its behavior in highly mobile networks. In the next chapter we will address this problem and show that it is possible to eliminate the impact of very high node mobility on the performance of position-based multicast routing.

CHAPTER 5

Contention-Based Multicast Forwarding

The main problem of SPBM, presented in the previous chapter, is its behavior in highly mobile networks. Outdated neighbor tables cause failed packet transmissions, which compromise the packet delivery ratio. Thus, in this chapter we propose Contention-Based Multicast Forwarding (CBMF) [39, 183]. In CBMF, the candidate next hops determine the best-suited next hops by contention. The winners of such a contention will be the nodes providing the highest geographical progress towards the destinations. This principle helps to overcome the problems with outdated neighbor tables and makes the CBMF protocol resilient to node mobility.

5.1 Introduction

In this chapter, we propose a novel approach to position-based multicast forwarding in mobile ad-hoc networks. Like SPBM in Chapter 4, it is designed to be scalable with respect to group and network sizes. We will show by simulation that it also copes perfectly with node mobility.

Multicast routing requires that group membership information be available to at least some of the nodes. In Section 4.2.1 we described a group management scheme for position-based multicast that provides this information in a highly scalable way. Thus, we base our novel forwarding algorithm on this group management scheme.

A few years ago, Contention-Based Forwarding (CBF) [71] was proposed by our team in Mannheim as a novel way to perform position-based unicast forwarding in MANETs. It abandons the periodic sending of beacon messages containing node positions that is used in most other MANET protocols and transfers the choice of the best-suited next hop from the forwarding node to its neighbors. The neighbors compete by means of timers to be the chosen next hop: Upon packet reception, every node computes a time value based on the forwarding node's position, the destination's position (which are both stored in the packet), and their own position. The better suited a node is, i. e., the more progress it yields to the destination, the lower will its computed timer value be. The node whose time runs out first will forward the packet and suppress the other neighbors.

Since this algorithm does not rely on beacon messages for the retrieval of positions of neighboring nodes, its forwarding decisions are always based on the current positions of the participating nodes. It will not select neighbors which have moved out of range between the time when the last beacon was sent and the time when the forwarding takes place, nor will it miss neighbors that have moved into radio range since the last beaconing. This makes it much more robust against node mobility.

In multicast routing, a packet may require to be sent to more than one next hop. Since CBF uses broadcast to forward packets and since usually more than one neighbor receives the packets to be forwarded, we want to exploit this property for potential use in multicast forwarding. The idea is that a packet is sent out once, and the receiving nodes then decide which *subset of them* will forward the packet.

5.2 The Contention-Based Multicast Forwarding Algorithm

Contention-Based Multicast Forwarding is a forwarding algorithm that is based on the location and group management scheme described in Section 4.2.1. It may be used alternatively to the forwarding algorithm defined by Scalable Position-Based Multicast.

To recapitulate: The group management scheme of SPBM divides the area of the ad-hoc network into different levels of squares, using a quad-tree structure. The network area, which all nodes must be aware of, is divided into four level-($L-1$) squares. Each of these level-($L-1$) squares is then divided into four level-($L-2$) squares, and so on. This is continued until the resulting level-0 squares are small enough that each node is able to communicate with all the other nodes within the same level-0 square directly, i. e., they are within radio range.

The group management provides every node in the ad-hoc network with aggregated group membership information for a number of squares. Each node is aware of the group memberships of all three neighboring squares on each of the levels. How this information is actually provided has been described in detail in Section 4.2.1.

5.2.1 Contention-Based Forwarding

The forwarding of CBMF uses the basic concepts of the Contention-Based Forwarding protocol (CBF) [116, 70, 72, 71], a unicast routing algorithm for mobile ad-hoc networks invented by our team in Mannheim. Following CBF, several other contention-based unicast routing protocols were proposed [81, 82, 200, 199, 34, 31, 30, 32, 192]. These all have in common that they use the geographic progress provided by the candidate forwarding nodes for a contention. In the following we will focus on the details of CBF.

In position-based unicast routing, the destination is a single geographical point. The mapping of a destination node to a geographical destination position is done by means of a location service. See [135, 67] for an overview of location services for position-based unicast routing. In the following, we will focus on the actual forwarding procedure.

The CBF algorithm consists of two major parts: the *forwarder selection* determines the next hop best suitable to forward a packet, and the *suppression* tries to minimize the chance that more than one hop is selected as the next hop. In contrast to common position-based routing [135], the forwarder selection is not performed by the current forwarding hop but by the candidate next hops themselves. This is achieved by means of backoff timers that are dependent on the progress a node is able to offer to a packet. On reception of a packet that has to be forwarded, each candidate next hop starts a backoff timer before forwarding the packet. The better suited a node is, the lower will its timer value be. This has the effect that the best-suited node will win and thus forward the packet, while the others overhear this transmission and may cancel their backoff timers.

[116] defines the backoff timer duration according to the following function:

$$t_{\text{backoff}}(p) = T \cdot (1-p) = T \cdot \left(1 - \frac{\sqrt{(d_x - n_x)^2 + (d_y - n_y)^2}}{\sqrt{d_x^2 + d_y^2}}\right), \quad (5.1)$$

where $\binom{n_x}{n_y}$ is the position of neighbor n, $\binom{d_x}{d_y}$ is the position of the destination, T is the maximum contention delay, and $p \in [0, 1]$. The maximum contention delay is a protocol parameter that has to be defined when implementing the protocol. Its setting creates a trade-off between a short forwarding delay and a delay that is long enough to allow a good suppression of contending nodes.

CBF defines three different suppression methods: *basic suppression*, *area-based* suppression, and *active selection*. The basic mode does not make any arrangements to avoid packet duplications, which may occur if the suppression fails to identify them on time. Especially if possible forwarders are far apart from each other, they might not suppress each other because they simply cannot receive each other's transmissions. The other two suppression mechanisms alleviate the chance of packet duplicates. In order to achieve this, the area-based scheme defines a forwarding area with the property that all nodes located in this area are within transmission range of each other. The maximum area that fulfills this property has the shape of a Reuleaux triangle [156, 157], which is the intersection of the circles around the edges of an equilateral triangle with a radius of the side length of the triangle. Theoretically, all nodes taking part in the contention are able to suppress each other, eliminating the chance of duplicate packet transmissions altogether.

The third suppression method, called active selection, is a two-phase approach. Contention is performed as in the basic scheme, but instead of forwarding the packet, the nodes only send an RTF (*ready to forward*) notification back to the previous hop, which may then actively decide on the next forwarding hop. Having chosen the forwarding hop, the previous hop sends a CTF (*clear to forward*) packet. This method is the one most effective with regard to duplicate suppression, but it introduces an additional delay and additional control packets.

Our new multicast protocol is based on the area-based suppression scheme. Figure 5.1 presents an example of this scheme. It shows a node N that has to forward a packet to destination D. All nodes located in the white Reuleaux triangle contend for selection as the best-suited node. Because of the geometric properties of the Reuleaux triangle, the nodes inside this area are all at a distance that is at most the radius of the radio distance. They are able to receive each other's packets and thus to suppress each other's forwarding of a packet. When contention starts, the nodes calculate a timer value according to Equation 5.1. The node that is located closest to the destination will win the contention, i. e., will have the earliest timeout and forward the packet, suppressing the others.

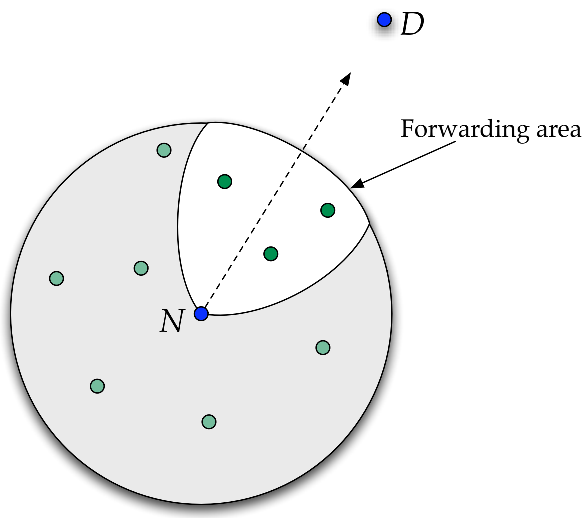


Figure 5.1: *The Reuleaux triangle in CBF area forwarding.*

5.2.2 Multicast Forwarding

In multicast, things are somewhat different. The destination for a packet that has to be forwarded is not a single geographic position, rather it is a list of square-shaped destination areas. If each of the destination squares is assigned a different Reuleaux triangle as a forwarding area, there will be as many different forwarding regions as there are destinations the packet is addressed to. In order to profit from multicast, i. e., transmit a packet to different destinations that are located in a similar direction only once, CBMF merges destinations according to the following heuristic: If the destination square is located to the north of the current node's square, the resulting forwarding region is the Reuleaux triangle that is oriented to the north. Similarly, this holds for squares in the south, east, and west. Destination squares that are located in between two of these directions are assigned to the corresponding diagonal Reuleaux triangles north-west, south-west, south-east, and north-east. Figure 5.2 shows the described predefined forwarding areas.

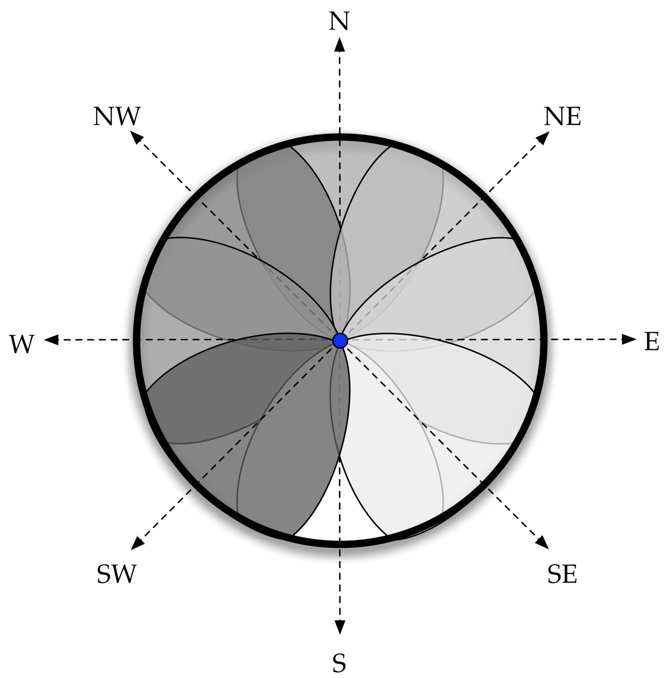


Figure 5.2: *The predefined forwarding areas in CBMF*

The complete multicast data distribution process works as follows: Before sending a data packet to a certain multicast group, a source locally looks up the squares that are currently subscribed to this group. It adds the squares to the packet header and broadcasts the packet to its neighbors. The neighbors receiving this packet are then able to determine whether they are located in one of the forwarding areas. This depends on their own position as well as on the positions of the source and the destinations. The positions of the destinations in relation to the position of the source determine the forwarding areas used. For all these areas the neighboring nodes then check whether they are located in one or two of the areas—since the areas overlap, it is possible that a neighbor may be located in two areas at once. In order to perform this check, a neighbor first calculates the corners of each area based on the source's position and the area orientation. Then it verifies whether its distance to the source, which marks one of the corners, and to the other two corners is less than the nominal radio range. If this is the case, the neighbor is located in the corresponding forwarding area. If a neighbor is in at least one of the areas, it will calculate the backoff time for each area it is located in according to the following equation:

$$t^i_{\mathit{backoff}} = T \cdot \left(\frac{i}{2} + 1 - \frac{d(S,F)}{r} \right), \tag{5.2}$$

where i is the number of the forwarding area. The forwarding areas in which contention will take place are numbered consecutively, starting from 1 for the first of the areas that is located in the north-west, and counting clockwise up to the area in direction west. Further, T is the maximum contention period, r is the maximum transmission radius, and $d(S,F)$ specifies the distance between the two nodes S (the sender) and F (the forwarding candidate).

The candidate nodes calculate and start their timers upon reception of a packet. During the first contention period the best-suited forwarder for the first forwarding area ($i = 1$) will be selected. Immediately after the end of the first contention period, the contention in the second forwarding area that is used (i. e., that has destinations in its direction) starts, and so on.

The difference between Equation 5.2 and the backoff function used by CBF is the following: While CBF uses the position of the destination node as a reference for the progress of a node, CBMF has to use a different reference; remember that packets are not addressed to a single destination that could be used as a reference. The approach of Equation 5.2 therefore is to calculate the backoff timer based on the distance of the current node from the sender of the packet; note that the node must be located in the correct forwarding region which implies that it provides a progress. A simple and intuitive way to accomplish this is to set the timer linearly depending on the distance—as done in Equation 5.2. However, this assumes that the nodes within a forwarding area are equally distributed over distance. The distribution of nodes in a Reuleaux triangle in fact corresponds to the arc

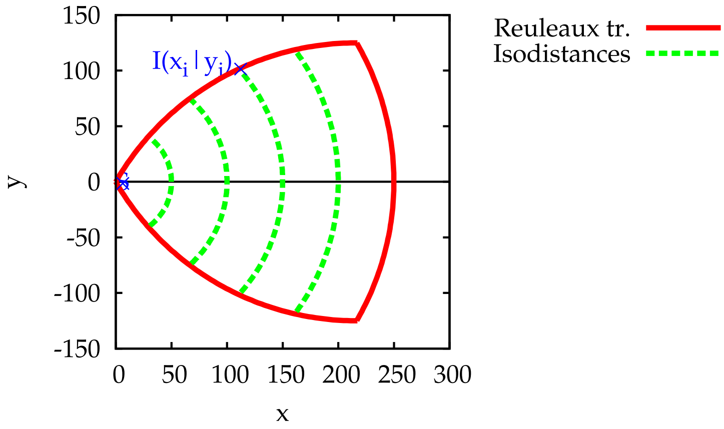


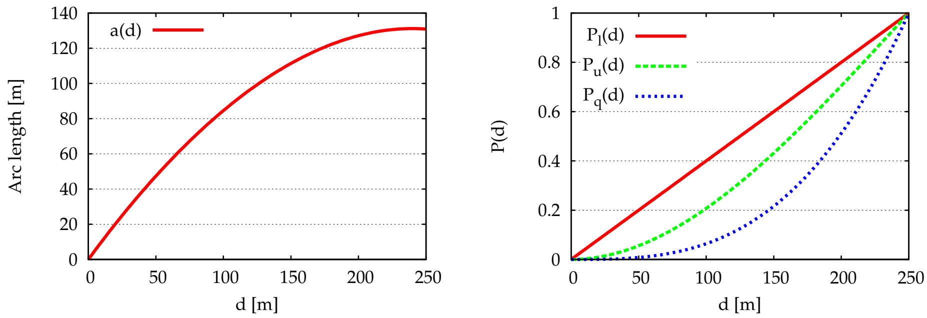
Figure 5.3: *A forwarding area in CBMF*

length of the circle describing equidistant nodes around the source node (see Figure 5.3):

$$a(d) = \int_{x_i}^{d} \sqrt{1 - \frac{x}{\sqrt{r^2 - x^2}}}\, dx,$$

where x_i is the x coordinate of the intersection points of the Reuleaux triangle and the circle with radius d. r, again, refers to the maximum radio range and thus to the radius of the Reuleaux triangle.

Figure 5.4(a) plots the arc length $a(d)$ with respect to the distance from the sender. If we assume that nodes are equally distributed over space, this means that the longer the arc is, the more nodes there are located at the corresponding distance. Since the backoff timer values are calculated based on the distance, the probability of a collision of two simultaneous transmissions by contending nodes is higher if more nodes are located at the same distance and thus share the same backoff value. In order to keep the collision probability low, the backoff timer function must take the probability of the nodes' distances into account. This probability can be described by the cumulative distribution function (CDF), which can be derived from $l(d)$ by means of integrating. We get $P_u(d)$ (for uniformly distributed nodes), shown in Figure 5.4(b). The linear backoff timer function assumes the CDF described by $P_l(d)$ in the same figure. The fact that the gradient of $P_u(d)$ is smaller than the gradient of $P_l(d)$ for distances less than approximately 100 meters means that the linear function accounts for more nodes than are actually present in a network with uniformly distributed nodes. In contrast, at greater distances fewer nodes are assumed. Nodes at greater distances provide a greater progress for a packet and will thus calculate a smaller delay. In effect, the linear function leads to more nodes per time interval for nodes with greater progress and thus to a higher collision probability for those nodes.



(a) Arc length with respect to distance from the sender

(b) Different CDFs of the nodes with respect to the distance from the sender

Figure 5.4: *Investigating the backoff function*

An alternative to the linear calculation of timer values is the assumption of a quadratically increasing distribution. This bears the advantage that the timer value will depend on the square distance. Since distances between two geographic locations are based on the Euclidean distance, the use of square distances saves the step of extracting the root in order to retrieve the distances. Especially for small devices like sensor boards that do not have a high CPU capacity this can be a good idea.[1] The node distribution that is assumed when using the square distances as a basis for the delay

[1]On a Texas Instruments MSP430 processor, the duration of the extraction of a root from a number around 100,000 varies between 67 ms and 75 ms, depending on the actual number. These delays and their variance are much too high for the use in a linear backoff function [118].

calculation is given as $P_q(d)$ in Figure 5.4(b). The quadratic delay function grants an even greater part of the backoff interval to the more distant nodes. This is reasonable because in dense networks there will be many nodes at great distances that will compete to forward a packet, while in sparse networks—if there is no node at a great distance—only few nodes will compete to forward at short distances.

Taking into account that a quadratic delay function has both better suppression properties and less computation complexity, we change the delay function given in Equation 5.2 to:

$$t^i_{\textit{backoff}} = T \cdot \left(\frac{i}{2} + 1 - \frac{d(S,F)^2}{r^2} \right).$$

Improving the Timer Efficiency

Having multiple contention periods for each hop on the forwarding path from the sender to the receivers induces a high end-to-end delay. For contention-based unicast forwarding in CBF, the authors elaborate on the trade-off between a low forwarding latency and a better resolution of the backoff timers, leading to a more efficient suppression of contenders [71]. In the following, we will present a method to reduce the maximum and expected delay for CBMF.

The time between the answer of the forwarder in one area and the start of the next contention period is unused. Especially if there is a well-suited forwarder for one area, it will answer at an early point in time within the contention period, thereby making the rest of the period dispensable. But since it may be the case that not all nodes that are waiting to take part in the following contention periods are aware of this early answer, it is not possible to shift the beginning of the following contention period to an earlier point in time.

The idea is now to split the contention periods into two parts. Instead of performing a complete contention period for each of the different forwarding areas, at first only the outer halves of the areas contend, reducing each single contention period to 75 % of the duration. The outer half of a contention area covers about 68.8 %[2] of the area of the complete Reuleaux triangle, and the best-suited forwarding candidates are located there. However, if for any of the contention areas no suitable forwarder could be found, an additional contention period will follow. During this period all nodes that are located in the inner parts of the Reuleaux triangles compete for medium access. Since these nodes are located within a circle whose radius is half of the nominal radio range, they are able to share a contention period. This contention period's duration is a third as long as the other preceding contention periods, the remaining 25 % of the original contention periods. Thus, the backoff timers can be computed according to the following equation:

$$t^i_{backoff} = \begin{cases} T * \left(\frac{3}{4}N + 1 - \frac{d(s,f)^2}{r^2}\right) & \text{if } d(s,f) < \frac{r}{2} \\ T * \left(\frac{3}{4}i + 1 - \frac{d(s,f)^2}{r^2}\right) & \text{if } d(s,f) \geq \frac{r}{2} \end{cases},$$

where N is the number of contention areas that are actually used for forwarding.

The inner area is used as a fallback for all outer areas. However, only nodes located in the respective Reuleaux triangles compete to forward the packet. Thus, it may happen that more than one next hop has to be chosen. Nodes that are located in the inner half of the radio range circle around a sending node are always located in two or three Reuleaux triangles at the same time. Thus, the worst case in terms of the number of forwarding operations within the inner area is four, i. e., four next hops within the inner area are able to cover all eight forwarding areas. If this worst case occurs, a packet will have to be sent four times during the contention period of the inner circle. The maximum duration of a CBMF contention period for a single area is between 10 and 20 ms, depending on how the value is configured. (Section 5.3 will deal with the determination of the optimal value for this parameter.) The IEEE 802.11 standard [87] recommends a maximum size for packets of 400 bytes when using a basic rate of 1 MBit/s for broadcast, resulting in a maximum transmission duration of 3.5 ms. Accordingly, there is enough time to forward multiple copies of a packet during the contention

[2] See Appendix A for the calculation of this value.

period in the inner area. Simulations showed that it is not necessary to halt contention for further forwarding directions during the transmission of the first packet.

The introduction of two phases for the contention periods significantly reduces the worst-case packet forwarding delay. Without this optimization, the worst-case delay is

$$t^{wc}_{\mathit{backoff}} = T \cdot N,$$

while the optimization leads to a worst-case delay of

$$t^{wc}_{\mathit{backoff}} = T \cdot \frac{3}{4}N + \frac{1}{4}.$$

E. g., if a packet has to be forwarded three times, and the maximum contention period is set to 20 ms, the optimization reduces the worst-case delay in the forwarding from 60 ms to 50 ms. If using the linear backoff function from Equation 5.2, the reduction will even be greater. Instead of 60 ms, the worst-case one-hop delay is then 40 ms, which is lower by one third than without optimization.

5.3 Evaluation

5.3.1 Simulation Setup

In order to evaluate the performance of CBMF, especially with respect to node mobility, we performed simulations with the network simulator ns-2 [5]. Since CBMF uses no beacon messages, it has an even lower simulation complexity in terms of generated packets than SPBM. Thus, the argumentation outlined in Section 3.3 does not hold for CBMF either and it is feasible to conduct the simulation experiments with ns-2. In the following we describe the scenarios and simulation parameters we used to simulate CBMF. To compare the performance of CBMF with that of SPBM, we also conducted the same experiments with SPBM. The MAC layer in all simulations was again IEEE 802.11 with a maximum bandwidth of 2 MBit/s. The transmission power was set to a value that yielded a radio range of 250 m.

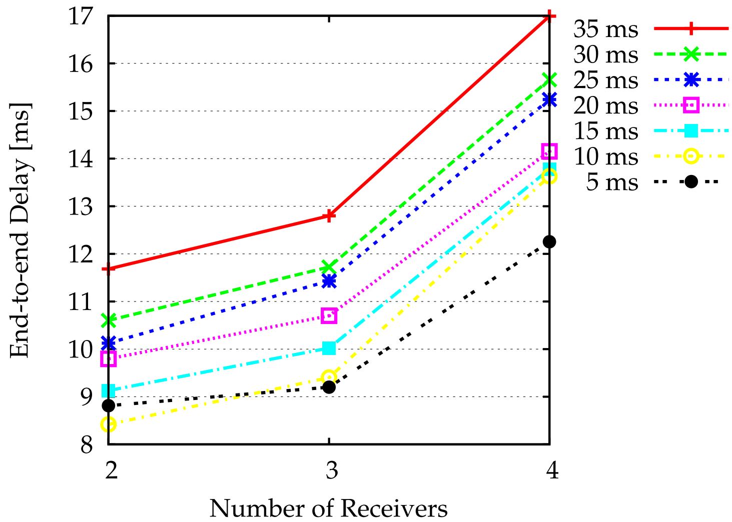
In the simulations with mobility, the nodes were moving according to the Modified Random Direction (MRD) mobility model [160] in a 3-level area of 1,000 m $\times$ 1,000 m. Note that the mobility model has no effect on the static scenarios. The difference between MRD and the Random Waypoint model (RWP) used in the previous chapters is the following: instead of picking a destination (RWP), the nodes pick a direction and a time how long to move in the chosen direction (MRD). Acknowledging the results from [26, 40, 197, 27] we decided to switch the mobility model. The duration of each simulation run was again set to 180 seconds. After a setup phase of 60 seconds, one sender was sending 60 packets to a group of 10 nodes, at a sending rate of 1 packet per second. The results presented below show the average over 10 runs with different random seeds.

In all simulations, CBMF used reactive group management as described in Section 4.2.1, while SPBM used proactive group management. Hence, the results for mobility include the reactively produced group management messages.

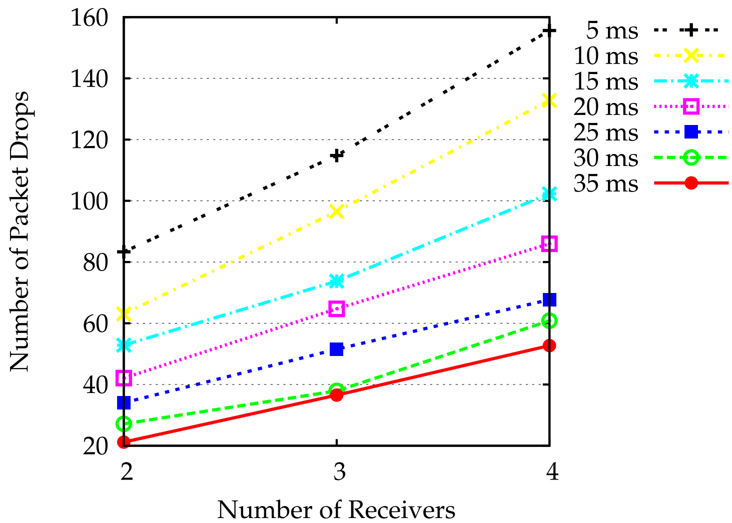
5.3.2 Determining the Optimal Contention Period

At first, we will determine a suitable value for the maximum contention period in CBMF. This parameter influences the end-to-end delay and the number of packet drops due to collisions during contention. There is a trade-off between a low delay and a low collision probability, both of which are desirable.

The results in Figure 5.5 show that at low values for the contention period, a variation of the value does not impact the end-to-end delay as much as it does at high values. The reverse holds true for the number of packet drops. Thus, a good value would be a medium contention period of 15 ms. A period of 10 ms would lead to many more collisions and thus to packet drops, while a period of 20 ms leads only to a slight increase of the end-to-end delay. The simulations described in the following sections will use a maximum contention period of 15 ms.



(a) End-to-end delay



(b) Number of packet drops

Figure 5.5: *Results for different contention periods*

5.3.3 Effects of Node Mobility

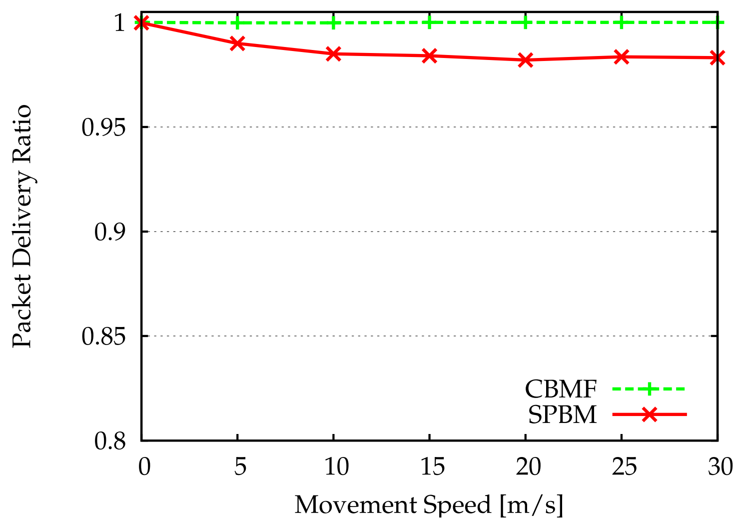


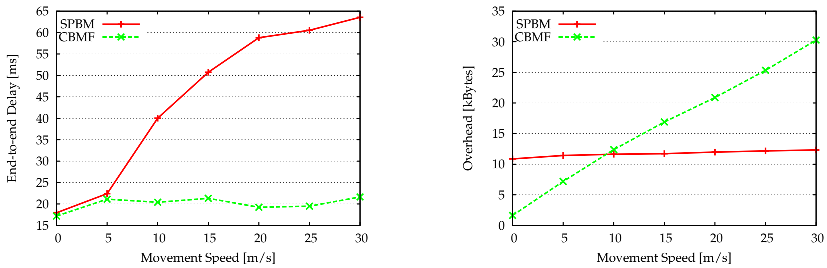
Figure 5.6: *Packet delivery ratio with respect to mobility*

The maximum node mobility in these experiments varied between 0 m/s and 30 m/s. The node density was set to 100 nodes/km^2, i. e., 100 nodes were simulated. Figure 5.6 shows the results for the packet delivery ratio. As can be seen, SPBM suffers a little bit from mobility, and the packet delivery ratio decreases to approximately 98 % at a speed of 30 m/s. However, CBMF is able to retain a perfect delivery ratio even at high speeds. This is due to two reasons:

(a) The contention-based forwarding is always based on the most current position information. Thus it does not suffer from stale or missing nodes in an outdated neighbor table.

(b) The reactive group management updates the membership information as soon as a node crosses a square boundary. Hence, all packets are routed to the squares where receiver nodes reside.

Figure 5.7(a) shows the average end-to-end delay of the data packets. The delay of SPBM increases with the mobility because of outdated neighbor



(a) End-to-end delay with respect to mobility

(b) Protocol overhead with respect to mobility

Figure 5.7: *Delay and overhead with respect to mobility*

tables that lead to unsuccessful MAC layer retransmissions of packets and finally to the re-addressing of these packets on the network layer. Although CBMF has a high worst-case delay for each single hop, the end-to-end delay is still lower than with SPBM.

Another very interesting metrics is the protocol overhead. SPBM has a constant overhead, nearly independent of the mobility. The update messages of the proactive group management and the announce beacons sum up to between 10 kB and 15 kB per simulation run. The reactive group management used with CBMF produces a linearly growing overhead through update messages that are sent when a group member crosses a square boundary. At speeds below 10 m/s, the reactive group management yields a lower overhead, at higher speeds the proactive management has less overhead, but this sacrifices the accuracy of the provided data and thus the overall performance of the routing protocol.

5.3.4 Effects of Node Density

The node density plays an important role when selecting a next hop. A higher node density means that the probability of a very well-suited next hop is higher, too. Figure 5.8 shows the packet delivery ratio for an increasing node density. The simulations were performed at a maximum movement speed of 10 m/s. At low densities, neither protocol is able to deliver a

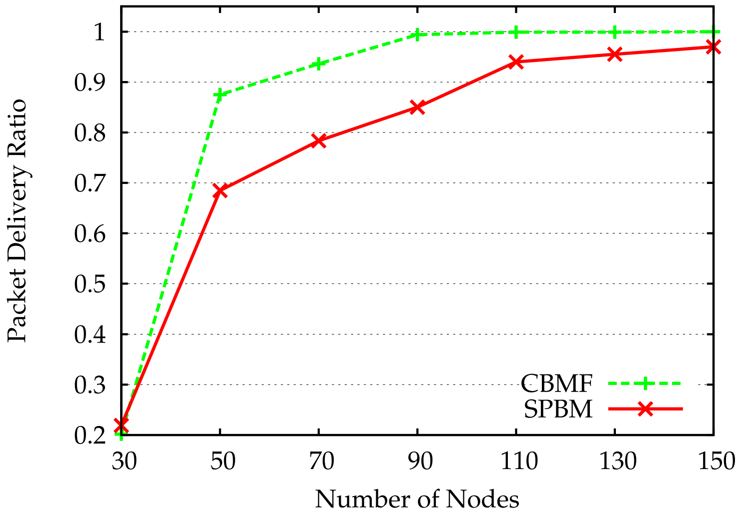


Figure 5.8: *Packet delivery ratio with respect to node density*

reasonable fraction of the packets. This is not surprising since the network is likely to be partitioned, and thus not all routes that are required for a successful delivery to all receivers exist. In scenarios with low to medium node density, CBMF delivers a significantly higher ratio of the packets than does SPBM. This is due to the fact that in these scenarios the average distance between neighboring nodes is higher. Thus, the probability to choose a node that is located at the border of the transmission range is higher. And in SPBM, that node could have moved out of the transmission range during the last beacon period. CBMF, however, alway uses the current positions and is able to profit from far-away nodes yielding a better progress. In high density scenarios, SPBM is able to catch up with CBMF.

5.4 Conclusions

In this chapter we presented CBMF, a contention-based multicast forwarding algorithm. It is based on the group management of SPBM (see Chapter 4), inheriting the scalability with respect to the group size. The main problem of SPBM is its behavior in highly mobile networks. With CBMF we addressed this problem and created a position-based multicast protocol that is independent of mobility. The choice of the next hops is transferred to the candidate next hops. They contend in order to determine the best-suited next hops for a packet. Simulations show that at any speed up to 30 m/s, all packets are successfully delivered. Furthermore, CBMF improves the routing in sparse networks by using the the most up-to-date position information available.

CHAPTER 6

Backpressure Multicast Congestion Control

In this chapter, we consider the problem of congestion control for multicast traffic in wireless multihop networks. We propose to apply a congestion control concept which is tailored to the very special properties of the wireless multihop medium: implicit hop-by-hop congestion control. The idea, so far only having been considered for unicast traffic, is here generalized to multicast. We implement it in the Backpressure Multicast Congestion Control (BMCC) protocol [165], with a focus on how to realize it in combination with geographic multicast routing in the Scalable Position-Based Multicast (SPBM) protocol (see Chapter 4). Our evaluation points out a number of highly desirable properties of the proposed scheme. In particular, it achieves and maintains high throughput and high packet delivery ratios at low packet latencies, even in the presence of significant network load.

6.1 Introduction

MANETs use a shared broadcast medium. All the nodes within a collision domain share the medium capacity, which is therefore a scarce resource. The multicast protocols presented in this thesis so far help to conserve resources when delivering data to multiple destinations. A shared broadcast medium, however, is also much more prone to network congestion than, for example, traditional wireline networks. Reducing the number of transmissions required to deliver the data to all receivers is therefore only half the battle. Especially in wireless multihop networks, it is also absolutely vital to efficiently control congestion to prevent a congestion collapse.

In this chapter, we propose a novel congestion control scheme for multicast in mobile ad-hoc networks. While generally exhibiting very competitive performance, it focuses particularly on the most demanding class of applications: those depending on very low packet latencies in combination with high packet delivery ratios.

Our protocol is based on *implicit hop-by-hop congestion control*, which is a paradigm introduced in the Cooperative Cross-layer Congestion Control protocol (CXCC) [164], but which thus far only has been considered for unicast traffic. CXCC combines simple packet scheduling rules and medium overhearing to implement backpressure with very short queues in the intermediate nodes. What we will describe here is how these concepts can be applied to multicast traffic. Since implicit backpressure is the key concept in our protocol, we call it Backpressure Multicast Congestion Control (BMCC).

We focus on an implementation of BMCC in combination with Scalable Position-Based Multicast (SPBM) (see Chapter 4). Implementing congestion control over such a scheme is particularly challenging: The source node has neither information on the continuously changing topology of the multicast tree nor on the number of group members. But BMCC not only works for SPBM, it can be used whenever a multicast routing protocol guarantees that forwarders know their set of next hop nodes in the multicast distribution tree. This holds for SPBM and also for a large variety of

other multicast routing approaches. Therefore, the ideas and concepts introduced here are not specific to SPBM. CBMF, however, performing much better in the presence of mobility, does not meet this requirement.
In our evaluation, we assess the performance obtained with BMCC using ns-2 simulations. We compare it to plain SPBM, to a variant of SPBM which is also introduced here, and to ODMRP [122]. The results of the simulations underline the very good performance of our approach.

6.2 Related Work

While multicast routing for mobile ad-hoc networks has received some attention over the last years (see Chapter 2), congestion control for this type of traffic in a wireless multihop environment has only been studied sporadically.
In [174] the MANET multicast protocol ODMRP is evaluated with a different MAC protocol than IEEE 802.11. Congestion control in that approach is performed in an end-to-end fashion. Explicit notifications inform the sender about the average load on the used links. The authors propose to use an explicit procedure (that further increases the load in the network) and argue that a backpressure mechanism would react too slowly. Our protocol, however, proves the opposite, reacting virtually immediately if the forwarding of a packet is delayed.
Similar to the above approach, Tang et al. [176, 175, 177] introduce an end-to-end congestion control protocol for multicast traffic. The authors propose to use negative acknowledgments to detect congestion. The sender reacts by reducing its rate until one affected receiver acknowledges a reception explicitly. Rajendran et al. [155] also use end-to-end rate adaptation. In addition, in anticipation of upcoming congestion they employ a local repair strategy that allows the retransmission of lost packets by intermediate nodes, thus reducing the amount of explicit congestion notifications. Both approaches, however, depend on feedback from the group members, generating a substantial amount of feedback traffic. Our protocol builds up backpressure immediately and locally, and avoids explicit feedback.

In [24, 25, 23], Baumung proposes a congestion-controlled multicast overlay for MANETs. Besides using a local recovery strategy to overcome packet losses, hierarchical aggregation of acknowledgments provides the source with feedback on the progress of the worst receiver. This feedback is then leveraged for congestion control. This approach is well-suited for overlay multicast, abstracting from the underlying network. However, detecting packet loss at the receivers and propagating the aggregated feedback may take a significant length of time. This is avoided by our approach. Peng and Sikdar propose a congestion control scheme for layered multicast in MANETs [150]. In their protocol, multicast layers are blocked and released in intermediate nodes, based on the observation of per-link output queue lengths and throughput measurements. Thus it is not possible to make adjustments finer than a whole layer. The scheme also does not take into account all aspects of the shared medium: Links are considered to be heterogeneous and lossy, but independent of each other in terms of capacity. The packets of a blocked layer are still delivered to the blocking intermediate node and dropped there. This wastes valuable shared medium capacity in the bottleneck area. These problems do not exist in our protocol. Substantial work exists in the area of MAC layer multicast in wireless environments. It does not deal with congestion control for multihop multicast traffic, but with delivering a packet to multiple (local) receivers. As one amongst other aspects, our scheme also has to address this question. A typical example is the Multicast MAC (MMAC) protocol [77]. In MMAC, the receivers of a transmission are listed in the packet header. Each of them acknowledges the successful receipt, in the order given by their index position in the header. In [93], a scheme is introduced which transmits a data packet to up to four receivers at once, and collects acknowledgments from them; for more than four addressees, clusters of at most four nodes are formed, and the packet is transmitted separately to each cluster. This paper also provides a broader overview of the area. We consider single-hop delivery to multiple addressees in a larger context, conjointly with multihop backpressure. This allows a different view of the problem. All previously proposed approaches result in significant control overhead, like, e. g., round-robin polling of all destinations, or many additional feedback fields. None of this is necessary in our approach.

6.3 Implicit Hop-by-Hop Congestion Control

BMCC extends the approach of implicit hop-by-hop congestion control to multicast. This concept has been introduced with CXCC [164], a cross-layer unicast congestion control protocol. Its key concept states that for each end-to-end connection an intermediate node may only forward a packet towards the destination after its successor along the route has forwarded the previous one. This creates a backpressure mechanism reacting very rapidly and effectively avoids excessive packet inflow into congested network areas. It hence recombines two functions traditionally located on the MAC and transport layers: single-hop reliability and congestion control. The concept is termed implicit hop-by-hop congestion control because there are no explicit congestion feedback messages and no dedicated mechanisms for congestion detection, nor are any windows or rates maintained. Congestion control itself is thus not an action that is performed by the nodes of the network, but an implicit effect of the packet forwarding rules. In CXCC, there are no multihop control packets at all.

CXCC obtains the information when a successor has further forwarded a packet by overhearing the channel (*passive* or *implicit acknowledgment*, see [102]). Until a further forwarding by the downstream node is heard, the respective connection is "blocked": No further packets originating from this sender and addressed to this receiver may be transmitted. This is depicted in Figure 6.1(a). Overhearing then serves two purposes at once: It constitutes an implicit acknowledgment, indicating the successful reception by the downstream node, and it also allows forwarding of the next packet. The final destination acknowledges the packet explicitly in that there is no next hop.

The overheard feedback on the successful reception of a packet by the next hop node may arrive with substantial delay. Immediate feedback is not required by CXCC, the protocol is built to deal well with such asynchronous feedback. This design principle is called *soft timing*.

However, wireless connections are potentially error-prone. The reception of a transmission by the successor along the route may fail, for example due to a collision, as it is shown in Figure 6.1(b). In CXCC, if a packet is not forwarded and thereby implicitly acknowledged after some time, a *Request For Acknowledgment* (RFA) packet is sent, a small control packet containing just enough information to identify the packet it refers to. Upon reception of an RFA, a node checks whether it has received the respective data packet. If not, it reacts with a *Negative Acknowledgment* (NACK), thereby triggering a retransmission. The RFA handshake avoids unnecessary payload retransmissions.

The transmission might also have been successful, but perhaps the implicit acknowledgment could not be overheard, like in Figure 6.1(c). The sending node cannot distinguish this situation from the previous one, and will send an RFA. The next hop node, however, is able to detect the difference. It then is aware that the implicit ACK has been lost. If no further (re)transmissions of the packet are pending and thus there is no possibility to acknowledge the packet implicitly later, this node will send an explicit acknowledgment to resolve the situation.

Finally, the forwarding may be delayed by contention or backpressure. In case of a very long such forwarding delay, an RFA might be sent for a packet. From a congestion control perspective, the reception of an RFA may then simply be ignored. Neither is a retransmission (triggered by a NACK) necessary, nor may the packet be acknowledged through an explicit ACK. This is undesirable since it would also release the backpressure.

If the receiver has moved out of range and the implicit acknowledgment is thus lost, the sender retries sending a predefined number RFAs before it notifies the routing protocol about the unreachable forwarding node. The routing protocol then determines how to proceed.

CXCC allows only one untransmitted packet to be queued in every intermediate node along the route. Note that this actually suffices not to waste any capacity: Since the shared medium enforces a local serialization of

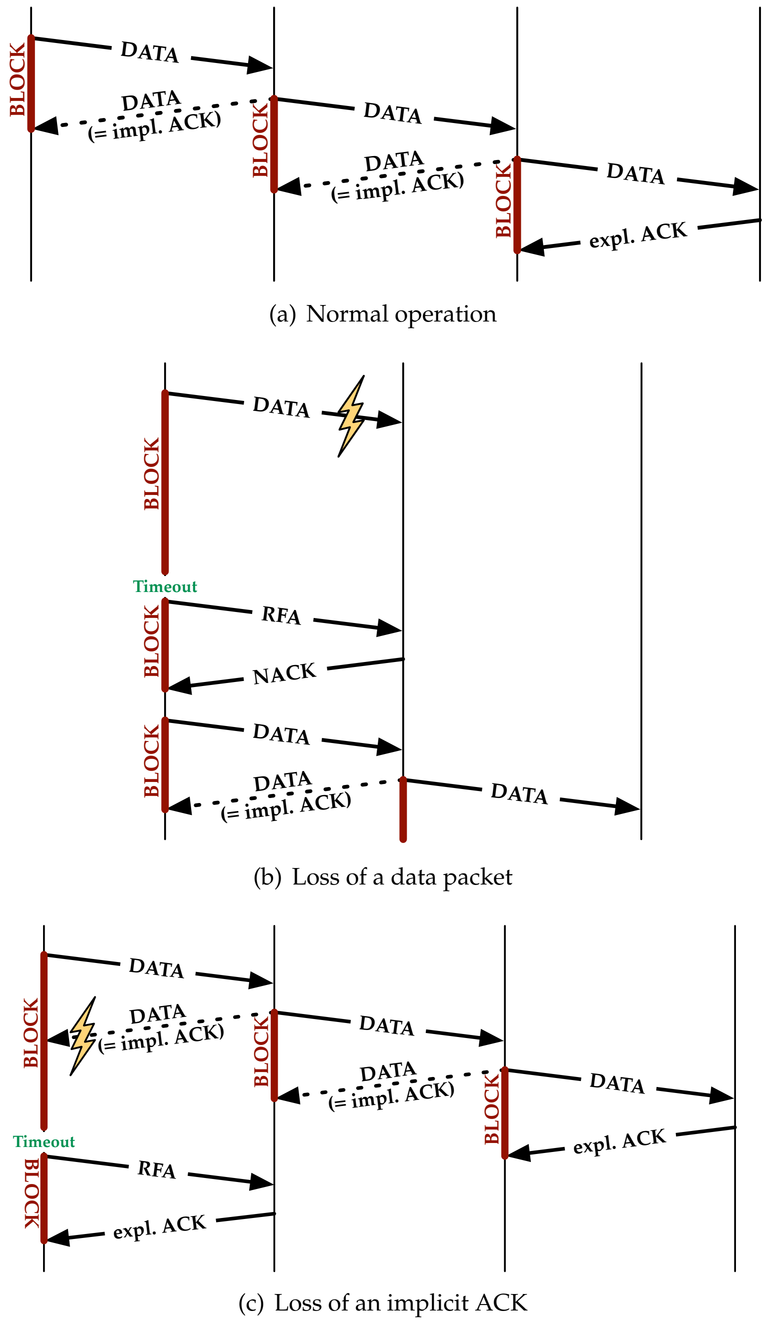


Figure 6.1: *Packet forwarding with the CXCC protocol*

transmissions, one packet per hop allows enough parallelism to make full use of the available bandwidth. Furthermore, CXCC does not increase the delay of the data stream since the first packet of a stream is forwarded as fast as without using CXCC and the remainder of the stream will use the full available bandwidth.

6.4 Backpressure Multicast Congestion Control

The central difference between unicast and multicast from the perspective of packet forwarding is that in unicast each forwarder has exactly one next hop node, while with multicast there may be more than one. Essentially, each packet is forwarded along a tree of nodes, originating at the source node. Therefore, in order to apply implicit hop-by-hop congestion control to multicast traffic, we need to generalize the implicit feedback concepts to that situation. This generalization forms the core of BMCC.

6.4.1 Packet Forwarding with Local Broadcasts

Implicit hop-by-hop congestion control exploits the local broadcast property of the wireless medium by not using explicit feedback, but instead gathering information through overhearing. SPBM as described in Chapter 4 uses unicast to accomplish the packet forwarding. Here, we follow the approach of transmitting the payload of a packet directed to multiple next hops only once by using MAC layer broadcast.

As a first step, we introduce a corresponding, alternative method of packet forwarding in SPBM, called Broadcast SPBM (SPBM-BC). The group management and the selection of the next hops are done as described in Chapter 4. But instead of a separate unicast transmission for each next hop, a single broadcast transmission is used for all of them. The forwarding node adds all designated next hops to the packet header, including for each a list of destination squares. When the packet is sent via MAC layer broadcast, all neighbors receive it and check whether they are contained in the list of designated next hops—if not, they discard the packet. This procedure is complemented by implicit acknowledgments: If the original sender does not overhear the retransmission of a packet from all of the designated next hops, it will rebroadcast the packet after removing all next hops from the header that have already successfully acknowledged the packet. This mechanism replaces the MAC layer acknowledgments and retransmissions of IEEE 802.11 unicast.

SPBM-BC is the basis for SPBM with Backpressure Multicast Congestion Control, but it will also serve as a benchmark: SPBM-BC will show us the performance that can be obtained by using local broadcasts and implicit acknowledgments *without* BMCC's backpressure mechanism.

To avoid parallel medium access attempts by multiple addressees that want to forward the packet, a node waits a random backoff time before it transmits. Multiple next hop nodes may not all be within mutual communication range, but it is reasonable to assume that they are often within carrier sense range. In combination with carrier sensing and medium access backoff, the jittering desynchronizes the responses, thus avoiding the synchronization problem. This solution matches the soft timing principle very well, avoids complex coordination, and saves significant overhead.

6.4.2 Backpressure with Multiple Next Hops

By not allowing the transmission of a subsequent packet before its predecessor has been forwarded by the next hop, CXCC builds up backpressure. This guarantees that a downstream bottleneck rapidly propagates backwards along the route towards the source. In BMCC, we apply the same concept, but along a tree structure. We strive for high packet delivery ratios to all receivers in this tree, i. e., towards all leaves. As a consequence, we need to adjust the source data rate to the tightest bottleneck in the forwarding tree. In other words, we need to ensure that the data inflow into any branch does not exceed the bottleneck capacity within that branch.

In this form, the scheme will be susceptible to the well-known "crying baby problem" [85]. If one group member has a particularly bad connection, this will result in a reduction of the service quality for all other group members.

We will devise a way to deal with this effect later. For now we concentrate on a backpressure protocol that adjusts the source data rate to the tightest point in the multicast tree.

BMCC achieves the desired congestion controlling behavior by generalizing the CXCC backpressure rule in the following way: The next packet may only be forwarded if *all* the next hop nodes for that packet have forwarded the previous one. Similar to the backpressure building up backwards along the route with CXCC, this rule in BMCC results in backpressure along the tree. Thereby, packets that are not able to traverse the network will not be allowed to leave the source node. This implicitly regulates the source data rate, and it keeps the queues in the intermediate nodes extremely short. Each forwarder can queue at most one untransmitted packet. The source node can also communicate the backpressure to the application. This allows to adapt the packet generation to the medium's situation, for example by adjusting the bit rate dynamically.

Since transmissions in BMCC are directed towards a set of next-hop nodes, the situation is significantly more complex than in the single next-hop case of unicast forwarding. Each single next-hop node may have received the transmission correctly or not. If the packet has been received correctly, each of the next hops may already have forwarded it again or might still have held it back due to backpressure. Finally, for each successor having forwarded the packet, the implicit acknowledgment may have been overheard or not. The central challenge in BMCC is to deal with this additional complexity efficiently while following the principles of implicit feedback and soft timing, and avoiding unnecessary control traffic.

To tackle this challenge, a forwarding node in BMCC keeps track of the list of next hop nodes from which an acknowledgment has not yet been received. After transmission of a packet addressed to a set of one or more next hops, this list is initialized to contain all these next hops. If an implicit (or explicit) acknowledgment from one of them is detected, the respective node is removed from the list. The transmission of the subsequent packet is not allowed until the last entry has been removed from the list.

If acknowledgments are missing for too long a time, a generalization of CXCC's RFAs is used. Analogous to data packets, RFAs in BMCC are directed to a whole set of next hop nodes: They address all the next hop nodes from which an acknowledgment is still missing. The addressed forwarders may then decide individually whether they should react with an explicit ACK or NACK.

A number of optimizations is possible to exploit the information contained in these handshakes most effectively. Since a single next-hop node that did not receive the data packet already necessitates a retransmission, it is not necessary to wait for feedback from all nodes if a NACK from this single node is received. Instead, an immediate retransmission of the data packet to the nodes from which acknowledgments are missing is triggered. Ideally, this makes the transmission of further NACKs by other next hop nodes unnecessary. Furthermore, such a retransmission may also fulfill the purpose of an RFA for nodes that have already received and forwarded the packet. If their forwarding has not been overheard by their predecessor, they will be addressees of the retransmission. They can easily detect this situation and repeat the lost feedback by means of an explicit ACK.

Like for the packet transmissions themselves, a possible synchronization of the answers to an RFA needs to be considered. If multiple addressees all access the medium immediately after receiving the RFA, this will cause severe collisions. For this reason, such reactions by forwarders, just like forwarded data packets, are sent with artificially generated jitter.

The design of BMCC is of course a trade-off. The protocol needs to keep track of the receivers from which no acknowledgment has yet arrived, construct RFA packets, etc.; this requires little, but not totally negligible storage and computational effort. However, BMCC is tailored to wireless multihop networks, and there the trade-off between computation power and communication bandwidth is very different from the situation in, for instance, high-speed Internet routers. We consider the—still limited—additional effort in the intermediate nodes appropriate because it helps to use the scarce MANET bandwidth more efficiently.

6.4.3 Dealing with Unavailable Next Hops

In order to perform effective congestion control, backpressure should be maintained as long as the downstream nodes are not able to forward the previous packet. However, indefinite waiting for an implicit acknowledgment from a downstream node which is no longer reachable must be avoided. Such a node will obviously not react to RFAs. But since this also applies to a node retaining a packet due to backpressure, a lightweight mechanism is needed helping to distinguish these two cases.

SPBM already provides a basic solution to this problem: If no more update beacons from a neighbor are received over some time, it is considered unavailable. But due to the relatively low beaconing frequency, this mechanism reacts rather slowly. In BMCC, we speed up the detection of no longer available next hops by using *keepalive (KAL)* packets. A KAL is a small control packet sent if an RFA is received for a packet which has arrived, but is currently being held back due to backpressure. It may also be sent immediately, i. e., without waiting for an RFA, when a new packet is received from the previous hop, but an acknowledgment for the preceding one has not yet been received. The KAL indicates that its sender is reachable, but it does not release the backpressure. With this extension, the link to a next hop node may be considered broken if the number of consecutive unanswered RFAs exceeds a certain threshold.

While at a first glance the additional feedback messages seem to increase the protocol overhead in a situation in which the medium's bandwidth is particularly scarce, they can in fact help to *reduce* the total amount of control traffic. The reception of a KAL indicates that backpressure definitely exists. Consequently, RFAs are then sent less aggressively, resulting in a lower overall network load.

6.4.4 Handling Inhomogeneous Receivers: Backpressure Pruning

One issue mentioned earlier still deserves attention: BMCC will adjust the data rate to the tightest bottleneck in the multicast tree, i. e., to the slowest receiver. While this is necessary in order to achieve high delivery ratios at all receivers—and might well be desirable in certain usage scenarios—, it is susceptible to the "crying baby problem". If there is one group member that is particularly hard to reach, this will thwart a higher data rate to all other receivers. It is thus of interest to see whether a variant of BMCC can be built that behaves differently in this regard: Is it possible to modify the algorithm to adjust the inflow into each branch of the multicast tree to the highest rate sustainable by at least one receiver in that branch, thus maximizing the throughput to each individual receiver? Depending on the application, the version described above or such a variant may be favorable.

However, due to the shared broadcast medium the rates to the receivers cannot be individually and independently maximized. For clarification, let us consider two simple examples in a scenario as depicted in Figure 6.2. There is one sender and two receivers. While receiver 1 is directly reachable from the source, receiver 2 is further away. In the first example, there is no additional traffic in the network. Transmissions from the source to receiver 1 are affected by transmissions made by at least the first two forwarders towards receiver 2, because of the shared medium and carrier sensing at the source node. BMCC aims at high packet delivery ratios to all receivers. The backpressure rule as presented above will achieve the following: It will always allow to forward a packet towards receiver 2 before the next packet enters the network—even though this reduces the throughput to receiver 1. Ideally, receiver 1 should not receive data at a higher rate if this comes at the cost of receiver 2's rate.[1] If aiming to achieve fairness among multiple receivers and high packet delivery ratios, this is generally the desired behavior.

[1]This is actually related to the notion of max-min-fairness, which states that a resource allocation is max-min-fair if increasing the share of any component is possible only at the cost of decreasing the share of an already lower component (for an in-depth discussion in the networking context see, e. g., [154]). We do not claim that the variant of BMCC to be introduced now will guarantee max-min-fair bandwidth allocations—due to the complexity and stochastic nature of a wireless multihop environment such a guarantee is hardly possible. But we aim for a heuristic that follows this general idea.

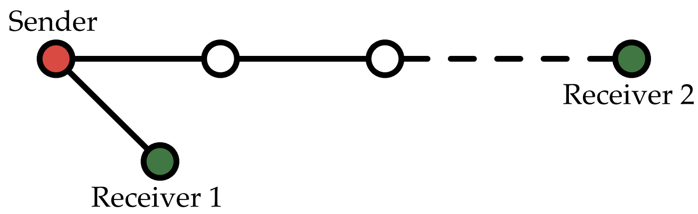


Figure 6.2: *Simple scenario with unequal receivers*

As a second example, let us consider a situation in which the medium around receiver 2 is severely congested. Then, backpressure towards the source will build up, and forwarding of packets along the route to receiver 2 may be substantially delayed. But this behavior can result in substantial *underutilization* of the medium around the source and receiver 1. Depending on the application, it may be desirable to use such otherwise unused medium bandwidth to forward additional packets to more easily reachable receivers. Nonetheless, only those packets should enter the network for which the bandwidth towards at least one receiver suffices.

We will now present a modification of the backpressure rule of BMCC that is able to yield just these effects. We call it BMCC with *backpressure pruning* (BMCC-BP). This backpressure pruning mechanism allows branches to be cut off if they contain backpressure. It makes use of the keepalive packets introduced above to improve detection of unavailable forwarders. Recall the conditions under which a KAL packet is sent: backpressure situations, when the forwarding of packets is delayed. Thus, the reception of a KAL from one next hop node indicates that the respective subtree is currently a bottleneck.

In standard BMCC, a node must wait for all next hop nodes to acknowledge the packet (neglecting, for simplicity of discussion, possible unavailable next hop nodes). BMCC-BP replaces this with a slightly more complex rule set as follows. A node may stop further attempts to deliver a packet to all next hop nodes if

1. at least one next hop node has acknowledged the packet,

2. a KAL has been received from all other next hops, and
3. a subsequent packet is already available for forwarding.

At the source node, the latter criterion is fulfilled if the application has already generated a subsequent packet which is waiting in the queue. In intermediate nodes, it holds as soon as a follow-up packet has been received from the upstream node. This may happen when the upstream node, in turn, has received at least one implicit or explicit ACK and KALs from all *its* remaining next hops.

The first backpressure pruning criterion guarantees that each packet will eventually arrive at at least one receiver: If one next hop node has acknowledged the packet, this implies that it has been forwarded into at least one branch. Packets will thus still not enter the network at a rate higher than what can be sustained by the fastest group members. The second criterion antagonizes the choking of the bandwidth in other branches by providing each next hop with a chance to access the medium and thus at least with an opportunity to forward a packet.

Backpressure pruning may result in situations where a node receives a follow-up packet before it has attempted to forward the previous one. In this case, it should drop the previously received packet (for which it has sent a KAL), and instead enqueue the newly received one.

Figure 6.3 shows an exemplary sequence diagram with a sender and two receivers for a situation where the described criteria apply. Receiver 1 is able to receive all data packets, while on the path to receiver 2 a packet gets lost. Although the sender does not receive an acknowledgment for the second data packet from the next hop towards receiver 2, it continues sending the third packet. Since it received a KAL from forwarder 1, it is aware that there is backpressure in the right branch and keeps on sending packets for the faster receiver 1. Upon receiving the third data packet, forwarder 1 will drop the second one and enqueue the third one instead. Note that the first data packet, which is currently being handled, will not be dropped.

Summarizing so far, BMCC, as originally introduced, is designed to result in an adjustment of the source data rate to the tightest bottleneck in the net-

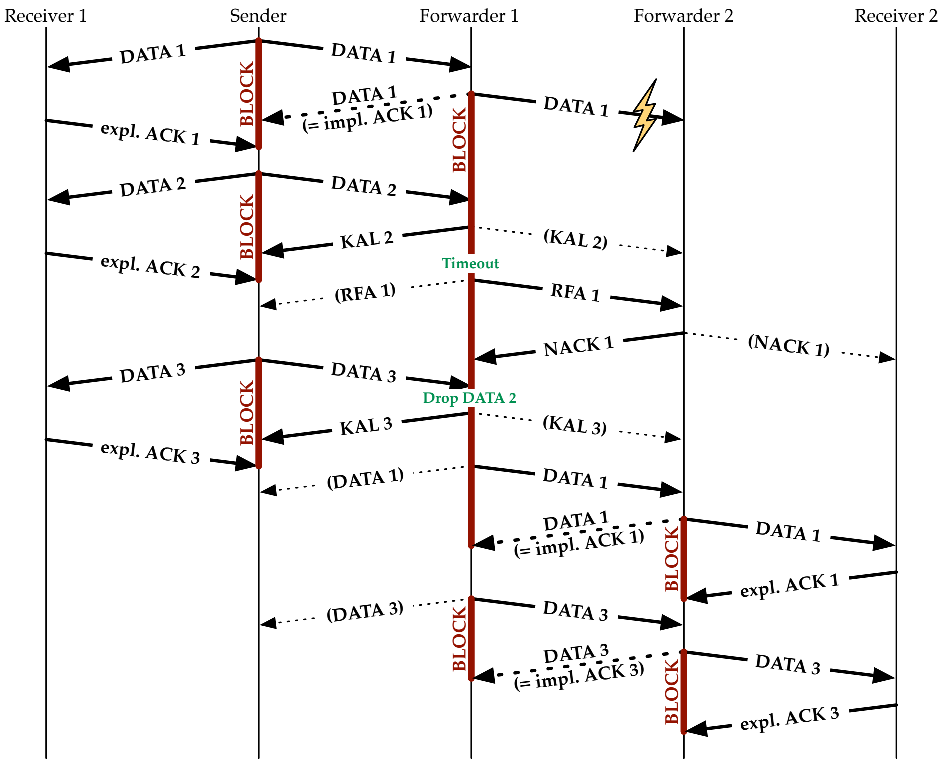


Figure 6.3: *Packet forwarding with backpressure pruning*

work. This ensures high delivery ratios whenever possible. BMCC-BP is in some sense complementary: It is built to deliver the maximum individually sustainable rate to each receiver, accepting that not all receivers will necessarily receive the same set of packets, as long as this does not come at the cost of other branches of the tree. Both adjust the rates of the source and those of intermediate nodes without explicit rate feedback and without multihop control packets, by using implicit backpressure.

6.5 Evaluation

For the evaluation we implemented SPBM with BMCC and BMCC-BP in the network simulator ns-2.30 [5]. As a comparison, we used the plain (unicast) version of SPBM as described in Chapter 4, the broadcast version introduced in Section 6.4.1, and an implementation of ODMRP [122] that was originally obtained from [6], ported to ns-2.30, and optimized as described in Section 4.3.1. For the results presented below, we simulated a network area of 1400 m$\times$ 1400 m with a total of 196 nodes (which corresponds to a node density of 100 nodes per square kilometer). Each data point is an average of multiple simulation runs in different scenarios. One multicast group was defined with two senders and ten receivers; thus, two independent multicast trees were used in parallel. We consider scenarios with and without node mobility. The source applications generate data packets with 64 bytes of payload at an increasing rate between 1 and 50 packets per second, or the highest frequency at which packets are able to leave the source node, whichever is lower. While BMCC can provide fine-grained feedback to the application about when packets may be sent, the other protocols used here are not able to generate such feedback. By accounting only for packets that are able to leave the source, we thus avoid distortions of the results for the other protocols and keep the comparison fair.

6.5.1 Delivery Ratio and Throughput

Shown in Figure 6.4 is the packet delivery ratio achieved by the different protocols in a static setting without mobility. A value of 1 means that all packets that leave the source nodes arrive successfully at each receiver. Adjusting the source rate in order to allow for high packet delivery ratios was a main design goal of BMCC. The results show that it has been achieved. At packet generation rates below 20 packets per second, the delivery ratios of unicast SPBM and ODMRP are very similar, with slight advantages for SPBM. For higher rates, ODMRP delivers a greater fraction of the packets. However, the fact that the delivery ratio continuously decreases for all three shows that more and more packets are able to leave the sources, but then do not make it to the receivers. Interestingly, the broadcast version of SPBM with implicit acknowledgments does not reach the performance level of unicast SPBM—although it theoretically needs fewer packet transmissions. Obviously, it is not enough to use implicit acknowledgments on the network layer. It is the backpressure mechanism in BMCC that turns the balance. It is able to outperform all others and reach delivery ratios very close to 100 % at all sending rates. This shows that the congestion control mechanism successfully regulates the source rate. The sending nodes only put on air as many packets as the network is able to deliver. Thus, the protocol is able to maintain high delivery ratios even for high packet generation rates.

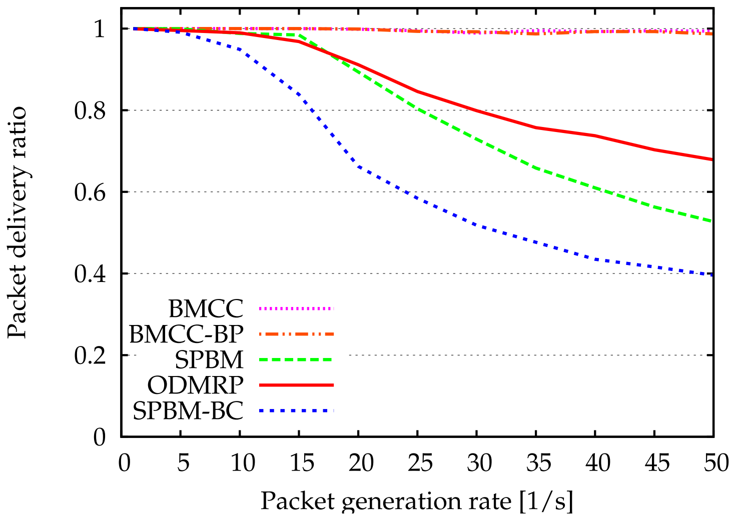


Figure 6.4: *Packet delivery ratio at an increasing packet generation rate*

In Figure 6.4, there is barely a visible difference between the results for standard BMCC and those for BMCC with backpressure pruning. This will likewise be the case in all our other random topology simulations. As we will soon demonstrate, the reason is not that the protocols generally behave identically; the effect is rather caused by the relative homogeneity of the receivers and therefore the load distribution in these networks. There seems to be virtually no free medium capacity that could be used to deliver data faster to certain receivers, without at the same time negatively affecting others.

Figure 6.5 shows how the packet delivery ratio develops in the presence of mobility. In these simulations, the nodes move according to the Modified Random Direction (MRD) mobility model [160], at a maximum speed of 5 meters per second and a pause time of 10 seconds. Data are again generated by the source applications at varying rates. It can be seen that the relative performance of the protocols remains largely unchanged, all deal reasonably well with moderate mobility—losing only a few packets. Rapid topology changes cause inconsistencies in SPBM's routing tables, and thus also affect SPBM with BMCC. At low packet generation rates, ODMRP slightly outperforms BMCC. But starting at 15 packets per second, the highest packet delivery ratio is consistently obtained with BMCC.

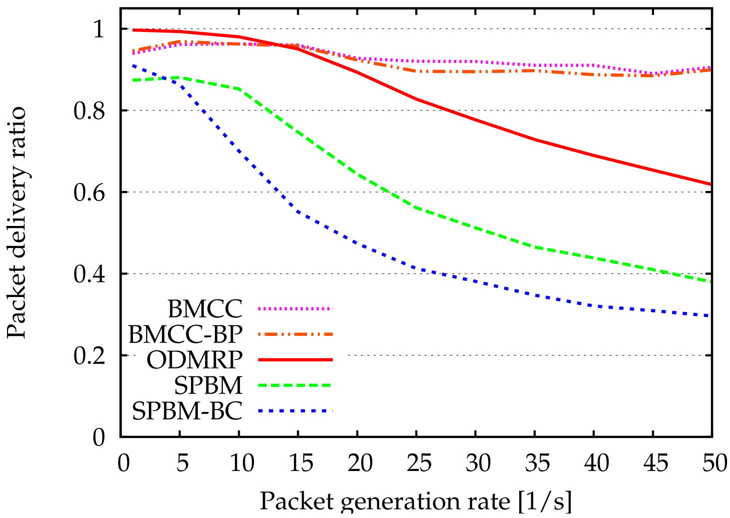


Figure 6.5: *Packet delivery ratio in mobile scenarios (maximum speed: 5 m/s)*

A very high packet delivery ratio could of course be achieved relatively easily if the total number of packets in the network is kept at a low level. Figure 6.4 only shows that almost all out of a so far unknown number of packets leaving the source do arrive with BMCC. We therefore have to consider these results in conjunction with the obtained data rate. Figure 6.6 presents the average data rate received by the group members. For packet generation rates of up to 15 packets per second, all examined protocols are able to deliver all the data produced by the applications. Since each data packet carries 64 bytes of payload, the resulting optimal data rate is 1,280 Byte/s at 10 packets per second from two senders. The simple broadcast version of SPBM (SPBM-BC) breaks first. Starting at 10 packets per second, its goodput increases much less than the data generation rate. Plain SPBM and BMCC show similar trends at different levels: the goodput grows up to a certain saturation and stays at the same level for all higher packet generation rates. While the unicast version of SPBM delivers on average around 2.3 kB/s, BMCC regulates the source rates to a level of 2.8 kB/s of goodput. ODMRP delivers higher data rates starting at 33 packets per second. This, however, comes at a high cost: Not only does ODMRP, as seen before, then lose at least 20 % of the packets. As we will soon see, it also allocates resources unfairly, preferring near-by receivers, burdens the network with a heavy traffic load, and suffers from high end-to-end delays.

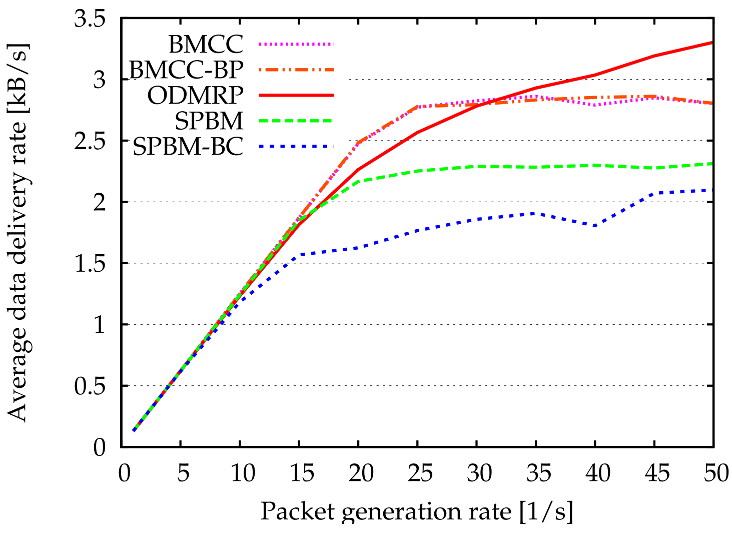


Figure 6.6: *Receiver data rate in static scenarios*

Figure 6.7 shows the average receiver data rate for mobile scenarios, i. e., the bandwidth at which data arrives at the receivers. Again, BMCC achieves a perfectly shaped throughput curve. As can be seen, ODMRP is nearly unaffected by mobility. The variants of SPBM, including BMCC, achieve lower rates in the presence of mobility. Starting from the point where ODMRP achieves higher data rates, it also—as described above—exhibits decreasing packet delivery ratios. In Chapter 4, it has already been shown that SPBM suffers from mobility because of its group management. Nevertheless, with BMCC, it is able to keep up high delivery ratios.

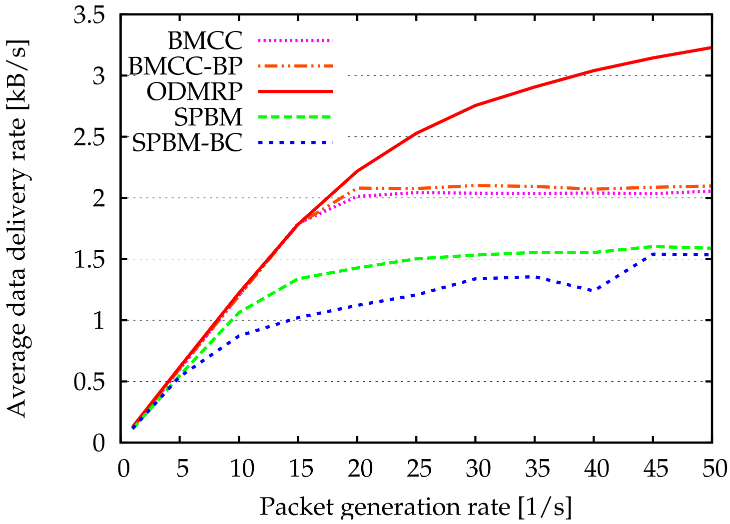


Figure 6.7: *Receiver data rate in mobile scenarios (maximum speed: 5 m/s)*

A considerable benefit of using a congestion control scheme like BMCC is that feedback on the sustainable rate to the application is possible. In contrast to approaches where a substantial fraction of the packets leaving the source node are lost in the network, this allows for an adaptation of the application behavior to the network's capabilities.

6.5.2 Fairness between Senders

The previous evaluation raises the question why BMCC does not achieve the somewhat higher data rates obtained with ODMRP if the network is seemingly able to support them. The key to understanding this property lies in the vastly different effort that is required to deliver a packet to different receivers, depending on their distance from the source. It is much more

resource intensive to bring a packet to a distant receiver than to one close by. A high data rate might simply be obtained by preferring transmissions over shorter distances. This issue is closely related to the fairness among senders: Do receivers preferably receive packets from closer source nodes?
In order to analyze this aspect, we look at the distribution of packet sources amongst the packets arriving at the receivers, with an increasing number of senders in a multicast group. To quantify the fairness of this distribution, we use Jain's fairness index as introduced in [92]. This index establishes a measure of the fairness of resource allocation in a multi-user system. It yields a value between 0 and 1, where 1 means perfect fairness and 0 is approached if one out of more and more participants is assigned all resources. The fairness index is defined as

$$\frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2},$$

where x_i is the resource share assigned to the i-th participant.
Here, we apply Jain's fairness index to the packet counts received from each source, thereby gaining a fairness value for each receiver. In our case x_i is the number of packets received from the i-th source. In Figure 6.8, the average of the resulting index values for all receivers is shown, for an increasing number of senders. ODMRP achieves better fairness than does plain unicast SPBM. Broadcast SPBM does not meet the performance of the unicast version in this metrics either. But BMCC again clearly outperforms ODMRP. Herein lies the reason why BMCC does not allow higher data rates: These seem possible only at the cost of increased unfairness.

6.5.3 Delay and Protocol Overhead

Another important metrics of protocols is the overhead incurred by their use. This is related not only to the efficiency of the medium use by the protocol, but also to the energy consumption caused by the transmissions. Figure 6.9 shows the average total amount of data that has been transmitted on the physical layer during one simulation run. ODMRP has the high-

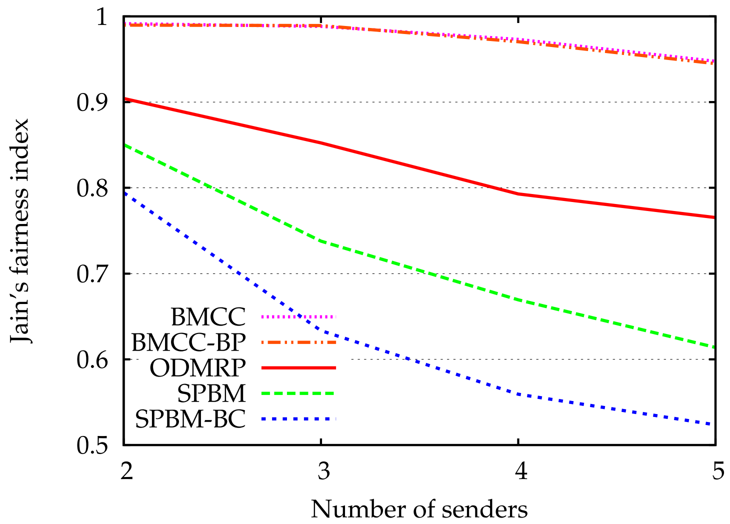


Figure 6.8: *Jain's fairness index for packet distribution over senders*

est resource requirements. The reasons lie in the structure of the protocol. ODMRP floods data packets through the whole network on a regular basis, and it uses redundant paths in a mesh structure, both of which result in a higher number of transmissions. The broadcast and unicast version of SPBM produce similar amounts of data on the physical layer. If local broadcasts are used with SPBM without employing BMCC's backpressure mechanism, even somewhat more bandwidth will be needed, instead of saved. This results from the high number of retransmissions performed in this approach. The backpressure mechanism of BMCC, because of its effective ways to avoid unnecessary retransmissions, is once again able to turn this into the opposite, avoiding unnecessary control traffic and retransmissions. One could ask why BMCC does not utilize more of the bandwidth (given that ODMRP is able to). But ODMRP only manages to put such high amounts of data on the medium because it regularly floods the whole network and thus also delivers data packets to nodes that are located in regions where no receiver is around.

Last but not least let us have a look at the end-to-end delay. Figure 6.10 depicts the average end-to-end delay of all delivered data packets, from

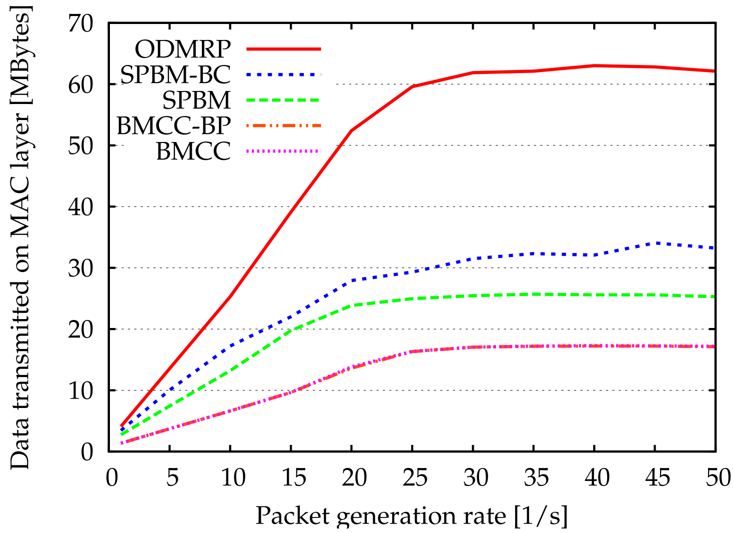


Figure 6.9: *Data transmitted on the physical layer*

the time the packet leaves the source node until the time it arrives at the receiver. Again, up to a packet generation rate of 15 packets per second in each source, all protocols deliver the packets with sufficiently short delays. At higher data generation rates, only BMCC is able to maintain short packet latencies. The other protocols delay the packets for up to two to four seconds, which is definitely unacceptable. This problem stems from long queues building up in the intermediate nodes, a problem avoided in BMCC by the very design of the protocol, which implies very short queues. Note that the unstable results for SPBM-BC at high packet generation rates stem from the fact that the protocol does not behave stable at these rates and the results thus show a high variance.

6.5.4 Backpressure Pruning

So far, it seemed that backpressure pruning does not have any noteworthy effect. This, however, is not true. The impression is a result of the relative homogeneity of the so far considered settings. To analyze the behavior of BMCC-BP in a scenario in which a difference must clearly appear, we use

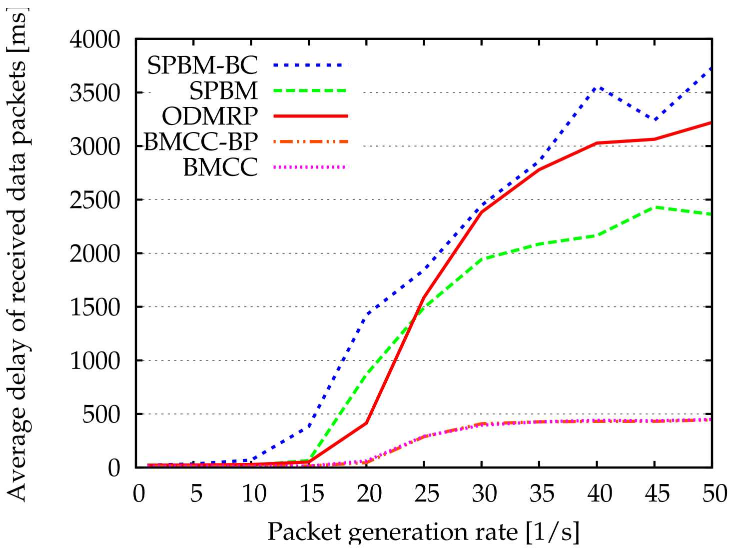
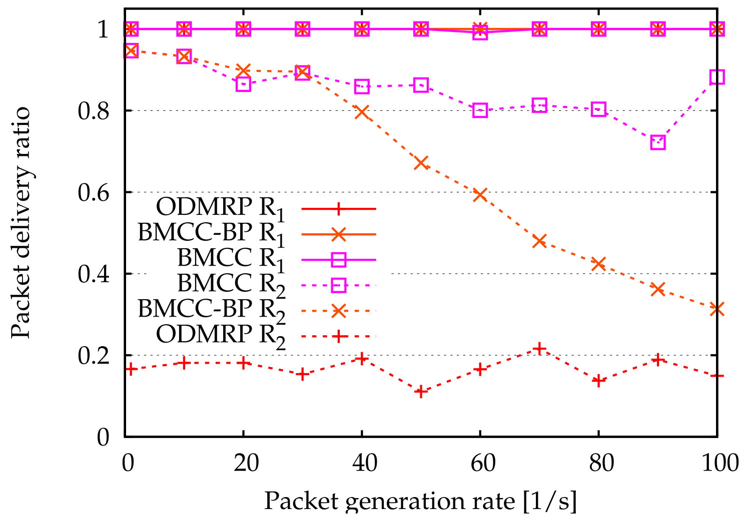


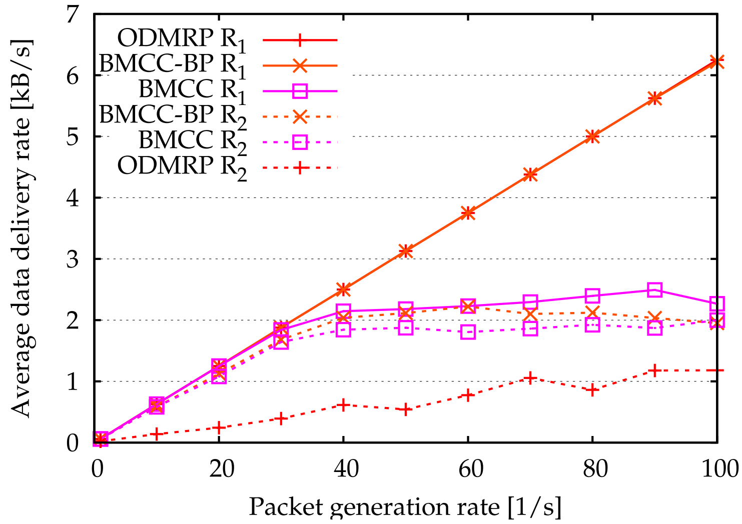
Figure 6.10: *End-to-end delay*

a simple static topology similar to the one depicted in Figure 6.2. Based on an equidistant chain topology, the nodes are set up such that the source is a direct neighbor to one receiver, R_1, while a second one, R_2 is seven hops away. An additional interfering data stream transmits packets continuously in the neighborhood of this second receiver. The source node again generates data packets at an increasing rate.

We analyze the packet delivery ratio as well as the data rate for each receiver separately. This allows a detailed analysis of the operation of BMCC-BP. The respective results are depicted in Figure 6.11. There are six curves, describing the results for ODMRP and for the two BMCC variants. For improved readability of the charts we omit the results with SPBM. Not surprisingly, all protocols are able to transmit packets with a high delivery ratio to the first receiver. BMCC, aiming at the maximum possible fairness, notices the congested area via its implicit backpressure mechanism and thus maintains a high delivery ratio also towards the second receiver—which is possible only at a limited rate for *both* receivers. ODMRP, lacking a mechanism to deal with such a congestion situation, results in a high number of packets lost on the path to the second receiver; the first receiver gets



(a) Packet delivery ratio



(b) Receiver data rate

Figure 6.11: *Results for receivers R_1 and R_2 in a simple static scenario with congestion at R_2*

packets at a high data rate, while the second receiver is generally cut off. The backpressure pruning mechanism in BMCC-BP handles the congestion situation correctly. It reduces the data rate to the second receiver (and thus the packet delivery ratio) without affecting the ability of the non-congested receiver to receive more packets.

6.6 Conclusions

In this chapter, we have proposed a novel way to effectively control congestion of multicast traffic in wireless multihop networks. Our scheme is based on implicit feedback, establishing multihop backpressure through simple packet forwarding rules. It solves single hop reliability and implicit multihop backpressure congestion control conjointly, thereby avoiding many unnecessary control messages and packet retransmissions.

A concrete simulation of the approach in combination with the geographic multicast routing protocol SPBM exhibits superior performance, demonstrating the effectiveness of the source rate limitation. Two flavors meet different demands: Adapting to the slowest receiver ensures that all group members receive the same packets, or supplying all receivers with as many packets as they are able to receive, accepting packet losses in the multicast tree if necessary.

In all cases, our scheme yields competitive throughput while maintaining very high packet delivery ratios for all receivers, combining these traits with very low end-to-end packet delays thanks to extremely short queues. It achieves all this at a small protocol overhead.

Chapter 7

Ad-hoc Multicast for the Real World

Having presented a number of protocols, it is time to show how they can be implemented for use in the real world. In this chapter, we will present two proof-of-concept implementations of multicast routing protocols for wireless networks. The first one is a Linux kernel module for Scalable Position-Based Multicast (see Chapter 4) [182]. It is designed for cross-platform usability on i386, as well as on ARM architectures, does not require any changes to the kernel sources, and offers a fully functional multicast protocol for MANETs on top of IEEE 802.11 [87]. The second implementation is Contention-Based Multicast Forwarding (CBMF) (see Chapter 5) for sensor networks [118]. It is based on the MSB430 sensor boards designed by FU Berlin and shows that the protocol is feasible even on resource-limited devices that especially do not provide any MAC layer.

7.1 Kernel Implementation of Ad-hoc Multicast

7.1.1 Introduction

Although interest in mobile ad-hoc networks has grown quickly in recent years, research in this area still strongly relies on discrete event simulation. However, recently published work challenges the value of simulations and forcefully argues in favor of real-world experiments [149, 187]. In this context, the number of real-world implementations for ad-hoc routing protocols continues to grow.

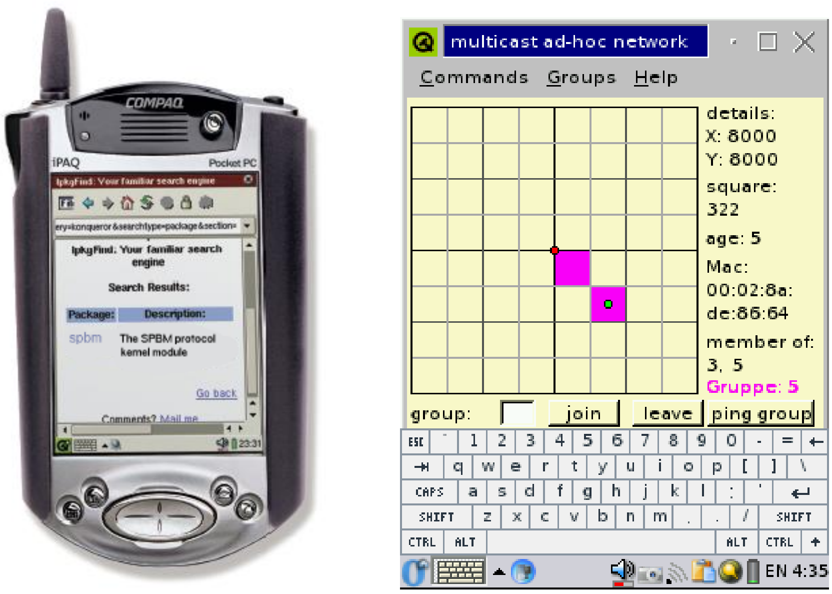
The advancing miniaturization and the availability of positioning systems allows the deployment of position-based routing algorithms [135] in mobile ad-hoc networks.

The first part of this chapter presents a real-world implementation of the position-based multicast routing protocol SPBM [182] that was described in Chapter 4. It is the first implementation of an algorithm of this class. Other implementations that were described in the literature include but are not limited to: a series of OLSR implementations [7], AODV-UU [129], AODV-UCSB [42], Kernel AODV [3], the FleetNet Demonstrator [79, 69], the CoRe project [2], and BLR [83]. A survey focusing on measurements with real-world implementations of ad-hoc networks is available in [107].

The C sources of our implementation are available from our web site [11]. They are compatible with Linux kernels 2.4 and 2.6 and do not require any modifications to the core kernel sources. Moreover, the module is prepared for x86 and ARM processor architectures. Figure 7.1(b) shows a screenshot of a GUI for the SPBM kernel module running on a Linux-based HP iPaq hand-held device (Figure 7.1(a)).

7.1.2 Preliminary Considerations

Before proceeding to the design details, we will briefly recapitulate the basic functionality of the SPBM protocol (see Chapter 4 for a more detailed description).



(a) HP iPaq device

(b) Screenshot of an iPaq GUI for the kernel module

Figure 7.1: *The kernel module also runs on HP iPaq hand-held devices*

The protocol can be divided into two main parts: *group membership management* and *multicast forwarding*. Either is based on a quad-tree structure, which is used to aggregate membership information by region, thereby allowing multicast forwarding to proceed in a hierarchical fashion. To use the protocol in a mobile ad-hoc network, each of the participating nodes has to be aware of the geographic dimensions of the network area. The quad-tree is then organized as follows: The entire network is divided into four equally-sized squares, each of which is again divided in turn into four sub-squares. This process continues until the diameter of the smallest squares is small enough to enable all nodes within the same square to communicate with one another directly. The subsequent splittings generate different "grid levels", and the number of splittings is referred to as "grid depth". Figure 7.2 shows an example of a network divided into a quad-tree of grid depth 3.

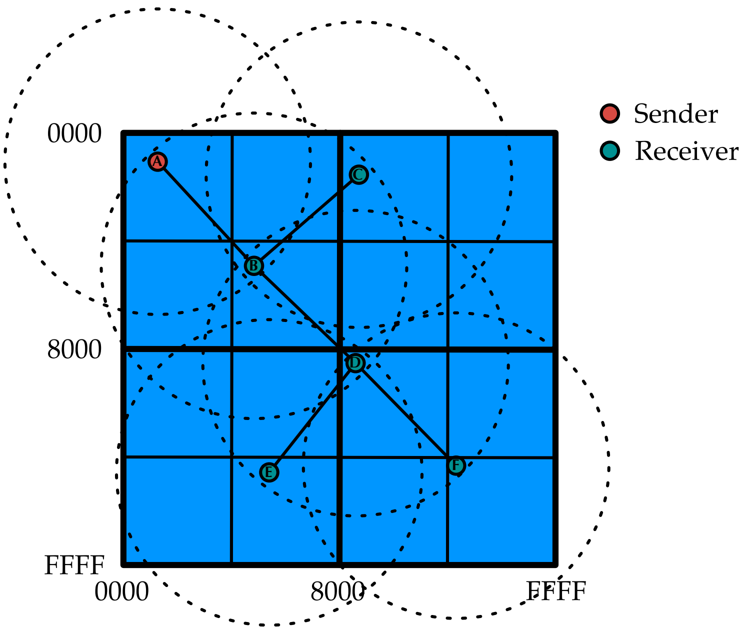


Figure 7.2: *Evaluation setup with six devices*

Information about multicast group memberships, i. e., the set of multicast groups in which a node is a member, is managed by means of bit vectors containing one bit per group (1 = subscribed, 0 = not subscribed). Each node periodically sends its vector to the nodes within its own lowest-level square. The disjunction of all vectors of one square constitutes the combined membership information for that square. This combined vector is propagated to the next higher level by sending a packet to all the nodes within the square on that level. The frequency of these propagation messages decreases with the level of aggregation, resulting in a very good scalability with respect to the size of the network.

Multicast forwarding is then done using the information collected by group management packets overheard by each node. When forwarding a packet, the forwarder stores the information about the remaining receiver nodes and/or squares inside the packet header. At the source, this is the information about all known group members, and at every consecutive hop,

the information about the set of remaining receivers. For the exact next-hop selection mechanism, please refer to Section 4.2.2. Roughly speaking, the forwarding and splitting process uses position-based routing towards node and square positions, relying on the fact that information grows more detailed as a packet approaches the final receivers.
Consequently, an implementation of SPBM has to comprise the following elements: First, it has to receive and send group membership update packets while managing the information about current memberships of adjacent nodes and squares; second, it must provide interfaces for incoming and outgoing data packets, as well as for data packets that have to be relayed; third, the protocol has to be controllable by the applications in regard to multicast group memberships; and finally, it has to implement a facility for obtaining the node's own position from a positioning device.

7.1.3 The Linux IP Architecture

In the following, we briefly point out how the Linux implementation of the IP protocol—as a representative of layer-3 protocols—works. This overview will serve as the starting point for the description of our implementation and is supposed to be of help in understanding the design problems and decisions that have to be taken during the implementation of a new ad-hoc routing protocol. A more detailed description can be found in [54].

Incoming Packets

On a Linux system, packets received from the network by an interface are differentiated on the basis of their layer-3 protocol type. In the case of Ethernet, this protocol type can be found in the header field ETH_PROTO_IP. Each layer-3 protocol provides a handler for incoming packets to which these are delivered. In the case of IPv4, packets are piped through a number of functions. Alongside checksum, lifetime and similar calculations and checks, the IP protocol's basic operation is the routing of packets. Depending on the destination address, a packet is delivered to the higher protocol layers at the local host, or it is handed out to the forwarding functions of the IP protocol. The forwarding part is responsible for IP options, memory allocation for the outgoing MAC header, and fragmentation of those packets that are too large for the layer 2 via which they are going to be sent out. Finally, the packet is passed on to the outgoing network interface.

Packets addressed to the local host are defragmented as needed, and delivered to a raw IP socket or the designated transport protocol.

Outgoing packets

Packets created by the transport layer are handled by a different part of the IP implementation. It provides various functions for the receipt of locally generated packets. In general, the first step is to determine a route for each incoming packet, i. e., to decide on which next hop the particular packet should be sent. Furthermore, the packet has to be fragmented if it is too large for the designated layer 2, and an IP header has to be added to the resulting packet(s).

Netfilter Hooks

During the processing of packets in the IP protocol, there are different hooks to which a different protocol or process can attach filter functions. These five so-called netfilter hooks are defined at significant places in the protocol implementation. They are in detail: IP_PRE_ROUTING is called immediately after receiving an incoming packet from the network, IP_LOCAL_INPUT just before delivering a packet to the local transport layer, IP_FORWARD during the forwarding procedure, IP_LOCAL_OUTPUT when a packet is received from the own transport layer, and IP_POST_ROUTING will be called before an outgoing packet is sent to the network interface.

7.1.4 Implementation

Fundamental Design Decisions

When implementing a routing protocol in Linux, the first main decision is whether to implement the protocol in kernel or in user space. While user space programs are much easier to debug, every routed network packet has to be transferred out of the kernel into user-accessible memory and back again, leading to higher additional latencies and thereby decreasing the usability of the protocol. For this reason, the SPBM protocol was implemented as a Linux kernel module, accepting the potential difficulties for the development process.

The goal of our implementation is to enable UDP communication between a sender and a multicast group. In terms of network layers, the module resides on layer 2.5, i. e., above the MAC protocol but below the IP layer. Consequently, the SPBM packet header is located after the MAC header and before the IP header.

An important requirement for the easy deployment of a kernel module is its compatibility with a non-modified standard kernel. In our case, this has implications for the use of IP addresses. Standard IP multicast addresses (ranging from 224.0.0.0 through 239.255.255.255 in IPv4) are handled by the multicast routing implementation already included in the kernel. This cannot be circumvented without modifying the original kernel sources. Thus, we defined the valid address range for SPBM multicast groups as 10.255.0.0 through 10.255.255.255. A drawback to this approach is that applications on the host must use unicast sockets rather than multicast sockets [171] to send and receive multicast traffic. This forces one to employ a different means of initiating multicast communication. We decided to make use of the `/proc` interface, being a standard method of the kernel to provide easy-to-use kernel-to-user-space communication. Figure 7.3 shows the architecture of the implementation which we will describe in detail in the following.

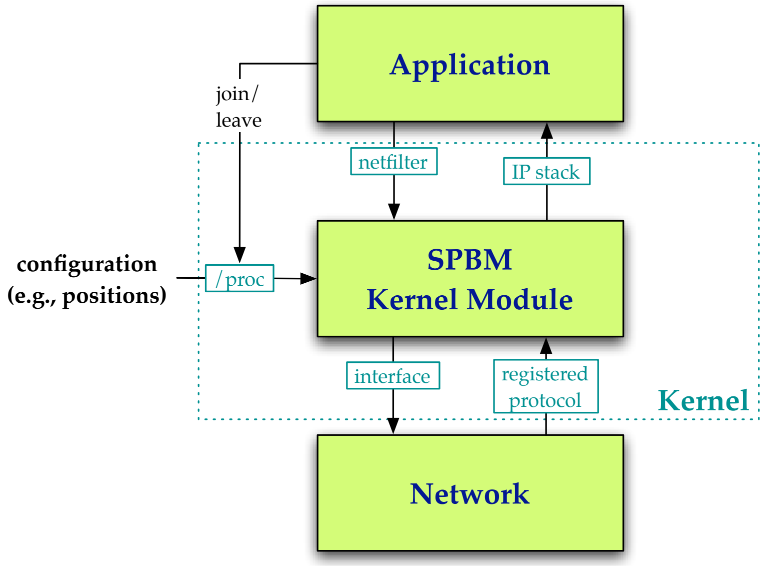


Figure 7.3: *Architecture of the Implementation*

Packet Handling

The basic mode of operation is as follows: Upon initialization the kernel module registers a new MAC packet type number for the SPBM protocol. Incoming packets comprising this type number are then automatically delivered to a handler provided by the module. This handler function is responsible for the further processing of the incoming packets: classifying them into control and data packets, changing the local state appropriately, and forwarding the packet if required.

Packets generated at the local host are filtered via the netfilter interface at the hook NF_IP_LOCAL_OUT and—if they are addressed to an SPBM address—captured by the routing module.

Outgoing packets are directly sent to the wireless network interface (device). In preparation, a buffer space is filled with an Ethernet header and the packet data. The Ethernet header contains the protocol type number of SPBM, the node's own MAC address as the source, and the next hop's MAC address as the destination.

While these techniques are common and likely to be used in many implementations of new protocols for the Linux kernel, packets coming from the network and addressed to the local host are handled in a way unique to SPBM. This is because we want to provide multicast communication via unicast sockets. The module deals with this issue by converting the received packets into standard unicast packets and handing them over to the IP stack. For this purpose, a new buffer space is allocated, MAC and IP headers are created, and the actual packet is appended. The IP stack itself is then able to deliver the packet to the application.

The group management part of the protocol requires that packets be sent at a certain time. Thus, a time-based event scheduler is necessary, which consists of a priority queue that holds the events and a kernel thread that processes them at the defined time. Separate threads yield two major advantages: First, an implementation bug occurring in an own kernel thread does not bring down the whole system. Second, it enables the routing module to serialize all function calls through the scheduler. Incoming packets and those waiting to be sent generate an event rather than being processed immediately. This provides a simple locking mechanism, allowing for conflict-free reading and writing of the SPBM data structures. Basically, the event list contains entries that consist of a time denoting the arrival of an event, a pointer to a function to be executed at event arrival, and a pointer to a socket buffer that possibly contains the packet to be processed.

Configuration Interface

As mentioned above, the `/proc` interface is used to configure the SPBM module. In order to enable applications to communicate with the kernel, `/proc` basically provides a virtual file system. Applications can read or write files in this part of the file system tree in order to communicate with the kernel.

If an application intends to initiate a multicast communication it writes the desired group ID to `/proc/spbm/join`, telling the SPBM kernel module to initiate a group join. To leave a group, a corresponding call can be made to /proc/spbm/leave. Reading from /proc/spbm/join gives a list of the currently subscribed groups. When communication is initiated by joining a group, a datagram socket to the corresponding IP address henceforth supplies the application with the multicast packets received from the group. Sending to this socket implies sending to the multicast group.

Another task of the configuration interface is to provide an easy way to inform the kernel about the current geographic position. To accomplish this, writing to /proc/spbm/position communicates the current position to the kernel, while reading from this file returns the current position of the node as perceived by the kernel.

Receiving Positions from a Positioning Service

Since the kernel itself does not provide trigonometric functions, the position format used on this interface (and in all protocol operations) complies to a two-dimensional Cartesian coordinate system using 16 bit per dimension, as opposed to the floating-point geocentric angular coordinate format provided by most of the currently available GPS systems. The process of reading coordinates from a positioning service, converting them to protocol coordinates and feeding them to the module has to be accomplished by a daemon specific to the positioning service used. It is evident that each participating node has to be configured with the geographic dimensions of the network in order to participate in multicast communication.

To make the kernel aware of the current position of the device, we employ a user space daemon. It has to regularly write its position to the virtual file /proc/spbm/position as 16-bit integer coordinates. A software product that may be used to accomplish this task is Loclib [108]. Amongst data from other positioning services, it is also able to parse NMEA 0183 messages from a GPS receiver. A customized version, that is available from the SPBM web page [11], translates the coordinates obtained from GPS into the coordinates required by the kernel module.

The usage of 16-bit integer coordinates introduces some position inaccuracy. Depending on the size of the network that is mapped to these coordinates, the maximum positioning error is $\sqrt{2} \cdot \frac{s}{2^{16} \cdot 2}$, which is about four centimeters for a network size of 2,000 m $\times$ 2,000 m, and thus, far smaller than the inaccuracy of GPS. Of course, handling networks of a much greater size could require positions to be encoded with greater precision.

7.1.5 Proof-of-Concept Deployment

To validate the correct operation of our implementation, we carried out simple tests with a setup of six nodes. In the experiment the nodes were "virtually" located as depicted in Figure 7.2 on page 136. In order to enable reproducible experiments, the nodes were physically positioned directly next to each other, with the topology being enforced by filtering packets from nodes with a virtual position beyond the transmission range, as depicted by circles in Figure 7.2. This setup causes an increase in the congestion level of the network since all nodes are in one another's interference range.

During each experiment we transmitted packets from node A to a multicast group that was joined by all other nodes. The sending rate of node A was limited only by the rate accepted by the MAC layer of node A, the size of the data payload was set to 1,000 bytes, IEEE 802.11 was set to 11 MBit/s, thus we have about 2.2 MBit/s gross for each link in Figure 7.2. We carried out the experiment 10 times. As a result, all nodes B through F, which were iPaq 3660 devices, received on average data at the rate of 408 kBit/s; no packet loss occurred. The latter was to be expected since there was no node mobility, and all transmissions of data packets were performed using unicast and MAC-level retransmissions.

7.1.6 Conclusions

We have presented a Linux kernel implementation of SPBM, a position-based multicast routing protocol. We have outlined how the protocol implementation module fits into the kernel architecture and how it communicates with the user space and the standard networking stack. Furthermore, we have shown basic protocol design components as well as a combination with the GPS positioning system. Finally, with a proof-of-concept deployment, we have demonstrated that the described implementation is working.

7.2 Multicast for Sensor Networks

7.2.1 Introduction

The term "sensor network" refers to a network whose main purpose is to communicate sensor readings or even more complex data, like photos taken by a camera connected to a sensor node. In wireless sensor networks, the nodes are equipped with a radio interface to communicate with each other. Some platforms employ the IEEE 802.15.4 standard [91], e. g., in the form of ZigBee [14]; others, like the ScatterWeb MSB430 sensor node [10] made by the FU Berlin, offer full access to the lower network layers.

Contention-Based Multicast Forwarding as it has been introduced in Chapter 5 was designed for MANETs. These are usually based on the IEEE 802.11 MAC layer [87]. Due to its timer concept, CBMF can easily be combined with a medium access protocol. It already provides most of the required functionality, such as medium contention and backoff timers. Since existing IEEE 802.11 hardware does not allow the protocols to be modified, we decided to implement CBMF for the MSB430 sensor nodes in order to investigate its usability for real-world applications when it is combined with a MAC layer.

7.2.2 Preliminary Considerations

Let us quickly recapitulate some properties of the CBMF protocol that have to be considered when thinking about an implementation (see Chapter 5 for a detailed description).

Based on a quad-tree, group membership information is managed in hierarchical squares. We will use the reactive group management as described in Section 4.2.1. It is particularly well suited for sensor networks where energy resources are limited because it refrains from sending periodic mainte-

nance messages and, hence, saves battery power. Only when a new member joins a group does this membership have to be propagated upwards through the quad-tree up to the level where group members are subscribed. Update messages are thus kept at a minimum.
Furthermore, our target is static networks. This means we will not need the movement prediction part of the reactive group management. Instead of validity timers that are dependent on the current speed of a joining node, we will be able to use static validity timers that do not depend on synchronized clocks.
Forwarding is based on contention. A sender broadcasts the packet within its radio range, and the nodes within radio range decide upon reception whether or not they will contend to forward the packet. This depends on whether they are located in one of the required forwarding areas. The size of these forwarding areas has to be meticulously chosen. For a good suppression of duplicate packet transmissions, a forwarding area should be small enough for a high likelihood of successful transmissions between two nodes within the same forwarding area. Furthermore, it should be large enough to provide a high progress for packets since only nodes within radio range will try to forward the packet. Thus, the size of the forwarding areas is in this case a protocol parameter like the contention delay. Both have to be chosen carefully for an optimal operation. We will do that based on measurements performed with the actual hardware platform.

7.2.3 The ESB430 Hardware

Microcontroller

The heart of the ScatterWeb MSB430 [10] sensor nodes is a microcontroller from the Texas Instruments MSP430 family: the MSP430F149. It features a 16-bit RISC CPU that offers 27 instructions and seven addressing modes. The controller is designed especially for ultra-low-power applications. It provides, among other things, 60 kB of EEPROM, 2 kB of RAM, and two clocks. One of the two clocks, the Auxiliary Clock (ACLK), is driven by a 32 kHz quartz crystal, which is also available in low-power modes for peripherals. The other, the Subsystem Master Clock (SMCLK), is fed by a digitally controlled oscillator with a frequency of 2.4576 MHz. The controller may be programmed in C or assembly language using a freely available toolchain based on the GNU Compiler Collection.

Timers

For the realization of the CBMF protocol, the timers are of central importance. The performance of the implementation will largely depend on the timer granularity. When programming the MSP430, two hardware timers are available that can individually be fed by either the ACLK or the SMCLK. Each timer is represented by a 16-bit register that is incremented at each clock tick. Depending on the chosen clock source, a timer tick takes either 30.5 μs (ACLK) or 0.407 μs (SMCLK). To generate a timer-driven interrupt, one of the *Capture Compare Registers* (CCR) has to be supplied with the desired timer value at which the interrupt should occur.
The two available hardware timers with a limited number of CCRs are not sufficient to implement the CBMF protocol, which needs timers for a series of purposes. Thus, we decided to use the concept of software timers, allowing a number of timers that is limited only by the available memory. Software timers are maintained in a linked list containing an entry for each timer. The first entry stores the timer ticks that have to pass before the first event is to occur. Further entries then contain a value that is relative to the preceding timer. Thus, on the one hand, during each hardware timer interrupt, only the first software timer value has to be decremented. On the other hand, it is much more complex to insert a new timer. However, it occurs much less often than the hardware timer interrupt. To implement the software timers, we will need only one of the hardware timers, and we can call it at any interval that is a multiple of one of the two hardware clocks.

Radio Transceiver

The ESB430 sensor board is equipped with an RF Monolithics TR1001 hybrid transceiver for short-range wireless communication. Its theoretical transmission range is 300 meters and more, but this transmission range is reached only under optimal conditions in open space. Because it operates in the ISM band at 868 MHz, interference with other radio equipment is common. The transmission power is adjustable between 0 and 99 on a non-linear scale. The TR1001 supports two different modulations: On-Off Keying (OOK) and Amplitude Shift Keying (ASK). OOK switches the transceiver on and off to transmit a 1 or a 0, respectively. ASK uses different power levels instead and thus is more robust to interference. Our implementation uses ASK and, in order to further increase robustness, a Manchester encoding that is provided by the ScatterWeb firmware. For our experiments we used a data rate of 38,400 Baud.

7.2.4 Implementation

Medium Access Control

The MAC used in our implementation is based on the ScatterWeb firmware. It has been considerably adapted for the use with CBMF. In the following we will line out how carrier sensing and collision detection work.

A MAC frame starts with a four byte preamble to allow the receivers to tune in to the tranmission. After a synchronization byte, two start bytes follow that indicate the transmission of a packet. Packet header and data are then sent in Manchester encoding. Concluding the packet, a two-byte Cyclic Redundancy Check (CRC) and a six-byte postamble are added.

Physically, the transceiver is not able to detect the sending of another node in range. But the synchronization byte, which consists of 8 bits that are set to 1, triggers an interrupt from the Analog-to-Digital Converter (ADC) at the receivers. Thus, these are aware that a transmission is going to start, and they are able to virtually detect a carrier. The following header then contains information on the length of the packet being transmitted. Based on this, the duration of the transmission is stored in a virtual carrier-sensing counter similar to the Network Allocation Vector (NAV) defined by IEEE 802.11 [87]. To decrement the virtual carrier sensing counter, we chose a main timer interval that approximately corresponds to the duration of one byte.

Since the transceiver has two different modes, one for transmitting and one for receiving, it is not able to detect a collision while it is transmitting. The node will in any case finish the transmission of a packet once it has started. The only way to detect the failure of a transmission is the absence of an acknowledgment that should have been (implicitly) received on the routing layer.

Memory and Computational Constraints

Memory is available in two flavors. While RAM is fast, its size is limited to 2 kB. It is used for data that changes during execution, like, e. g., packet buffers or membership tables. The available flash memory (EEPROM) has a size of 60 kB and can be written only about 100,000 times during its lifetime cycle. Furthermore, it may only be written in chunks of 512 bytes, which takes about 20 ms, and writing in flash is an exclusive operation, i. e., no other task can be performed in parallel. For these reasons, flash memory will store only the program code and variables that do not change their value during execution of the program, i. e., that are declared as "const".

This has implications, e. g., with regard to the number of packet buffers for packets in the receive queue and packets that currently take part in contention. It limits the number of flows that may cross the network at the same time. Furthermore, the stack should be kept small—i. e., as few stack variables and function calls as possible should be used.

The microcontroller contains a 8 MHz CPU. Floating point arithmetic causes especially high latencies and should be avoided wherever possible. This is accomplished by performing calculations based on square distances, superseding the calculation of square roots and, thus, floating point results. E. g., the floating point calculation of a distance between two positions takes up to 70 ms, the floating point calculation of the square distance takes about 5 ms, and the integer calculation of the square distance is done in less than 1 ms.

Another problem that arises from the weak CPU occurs if a node is located in two forwarding areas at the same time. Then it has to calculate the contention delay twice, which takes considerably more time than the calculation of the value for one contention delay at the nodes that are only located in one area. Since the time difference can be several milliseconds, such a node will have to subtract its additional calculation time from the resulting delay value in order to get a fair chance during contention.
Similarly, the check whether a node is in a given forwarding area requires the calculation of its distance from two reference points. Normally, if the first condition fails, the second will not have to be evaluated. But again, this would introduce a considerable imbalance if there are at least two forwarding areas to check. We thus have to ensure that both conditions are always checked. Replacing the operator "&&" by "&" in C performs this task and makes sure that both conditions are always checked.
Summarizing, one of the main challenges in the implementation is to give all nodes as equal a chance in contention as possible.

7.2.5 Measurements

Protocol Parameters

Setting the nominal radio range to an appropriate value is vital since timer calculation is based on this value. If the value of the radio range parameter chosen was less than the real distance, nodes farther away than this value would successfully receive the transmission and choose a timer of length zero. Hence, collisions would occur. However, if this parameter was chosen at a value larger than the real radio range, this would lead to a higher overall delay because there would be fewer nodes with small timer delays. Both effects should be avoided. Nevertheless, in this trade-off between delay and collisions, it is preferable to choose a more conservative and, thus, higher value for the radio range. In preliminary experiments we determined a value of 50 cm for the radio range in combination with the transmission power set to a value of 16, which suit the described requirements.

The second important parameter that influences contention is the maximum contention delay. It should be chosen as low as possible to minimize the overall delay but high enough to enable an effective suppression between nodes that are located close to each other. For the time taken to detect a carrier we measured a value below 112 ms in 98 % of the cases. Based on this value and the delay timer function, the collision range can be determined in which no suppression is possible at all. For a radio range of 50 cm, the measured 112 ms for the carrier sensing delay, and a maximum contention delay of 40 ms, we get a collision range on the order of 1 mm. Preliminary experiments showed that 40 ms is indeed a good choice for the maximum contention delay in order to achieve an effective suppression.

Test Scenario Setup

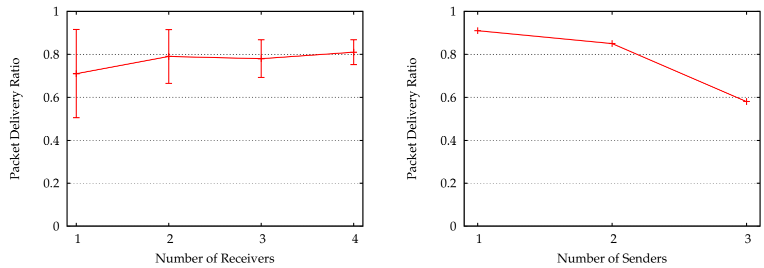
In order to evaluate the performance of our implementation of the CBMF protocol, we set up a grid scenario with 20 sensor nodes. With a radio range of 60 cm, the resulting scenario had a size of approximately 2 m^2 (see Figure 7.4). One of the nodes was declared to be a master node, responsible for the organization of the test runs. Before starting a run, the master node sends the values of the variable parameters, like packet size or sending rate, to all the other nodes. Then it schedules the synchronized start of the test run. After the run it collects the results from all participants so that they can be easily evaluated. In contrast to the test runs themselves, which were performed with a transmission power value of 16, the communication for the setup was done with the maximum possible transmission power. This ensured that no packet losses during setup occurred.
The positions used for the routing were predefined. Each node in the grid was assigned x and y coordinates according to its position in the grid. The positions were slightly jittered in order to avoid exactly identical distances in the grid, which would have led to collisions. The grid distance was chosen to be 50 cm so that a node in the grid was able to communicate directly with all its eight neighbors in horizontal, vertical, and diagonal directions.

Figure 7.4: *Test scenario setup with 20 sensor nodes*

Results

We conducted two experiments. The first one with a varying number of receivers (1 to 4) in one group, and the second one with a varying number of senders (1 to 3), each sending to a different group of three or four receivers. The packet size in all experiments was 32 bytes, sent at a rate of one packet per second; the number of retries in case a packet loss occurred was set to 2. Each experiment was run five times with different nodes that were sending and receiving. Figure 7.5(a) shows the average packet delivery ratio for the first set of experiments. It can be seen that the protocol's performance does not decrease when the number of receivers grows. The high standard deviation in the one receiver case hints that performance of the single receivers is quite different. If there are more receivers, this is statistically compensated, and the standard deviation shrinks.

The results of the second set of experiments is shown in Figure 7.5(b). The more senders there are sending their packets at the same time, the lower will be the achieved packet delivery ratio. The nodes have only a limited



(a) PDR with respect to the number of receivers (b) PDR with respect to the number of senders

Figure 7.5: *Performance of the CBMF protocol on sensor nodes*

buffer space for packets and thus are not able to store further packets that arrive from another sender when they are currently taking part in a contention.

7.2.6 Conclusions

In the second part of this chapter, we have described an implementation of CBMF for sensor networks consisting of ScatterWeb MSB430 sensor nodes. This includes the close combination with a MAC protocol and resource constraints that had to be taken care of. The results show that it is possible to use CBMF in sensor networks. Best performance is obtained when a single sender at a time is sending packets to a multicast group. Otherwise, data streams constrain each other because of the limited packet-caching capabilities in the sensor nodes.

CHAPTER 8

Conclusions and Outlook

Mobile Ad-hoc Networks offer a great possibility for research on a wide area of different problems that still have to be solved before MANETs can be deployed. One aspect is multicast routing protocols, which are required whenever a group of participants desires to communicate with each other. While topology-based multicast protocols had been numerously developed and thoroughly investigated, position-based approaches had been mostly disregarded. This thesis fills this gap and explores the potential of position-based multicast routing for MANETs.

8.1 Contributions

Prior to this thesis, only few approaches to position-based multicast for MANETs had been made. All of them had in common that the sending node had to precalculate the multicast tree over which the packets are distributed and store it in each packet header. This involves two main issues: (a) These approaches are not very flexible with regard to topological changes which abandons the advantages that position-based routing has against topology-based routing, and (b) they do not scale with the number of receivers, since every one of them has to be named in the packet header.

This thesis has started by solving the first issue, the flexibility. With Position-Based Multicast (PBM), Chapter 3 presented the design of a protocol following the forwarding principle of position-based unicast routing transferring the choice of the next hops in the tree from the sender to the forwarding nodes. Based on the positions of their neighboring nodes, these are able to determine the most suitable next hop(s) at the moment when the packet is being forwarded. Still remaining is the inaccuracy induced by outdated beacon messages and the location service which provides the source with the positions of the destinations.

The scalability with respect to the number of receiving nodes in a group is then solved by Scalable Position-Based Multicast (SPBM), introduced in Chapter 4. It includes a membership management fulfilling different tasks at once. First, it administers group memberships in order to provide multicast sources with information on whether nodes are subscribed to a specific group. Second, it implements a location service providing the multicast sources with the positions of the subscribed receiver nodes. And third, it geographically aggregates membership data in order to achieve the desired scalability. The group management features two modes of operation, a proactive one and a reactive one. It has been shown analytically that the overhead produced by the proactive variant is bounded and scales with $O(logA)$ with the size of the network. The reactive alternative, in contrast, reaches low worst-case join delays but does not limit the overhead. Overall, the scalability issue is solved, but possibly outdated beacon messages from neighboring nodes still affect performance in highly mobile networks.

Contention-Based Multicast Forwarding (CBMF), described in Chapter 5, then addresses the problems that appear in highly mobile networks. Instead of basing forwarding decisions on a perception that may no longer be up to date, the packets are addressed only to the final destination; no additional explicit next hops are specified. The receiving nodes, which are candidate next hops, then decide by means of contention which of them are the most suitable next hop(s) for the respective packet. Not only is the decision made based on the most currently available data, but this procedure also saves the regular sending of beacon messages, thus reducing the overhead. In the presence of mobility, the performance with CBMF is much better than it is with SPBM.

Having solved the principle problem of position-based multicast group management and packet forwarding, another unsolved problem obstructs high-bandwidth data transmission: the lack of multicast congestion control. Sending out more and more packets to a multicast group lets the performance decrease. The network should not accept more packets than it is able to handle. Backpressure Multicast Congestion Control (BMCC, Chapter 6) achieves this by limiting the packet queues on the intermediate hops. A forwarder may not forward the next packet of a stream before it has noticed—by overhearing the transmission of the next hop—that the previous packet has succeeded. If there is congestion in an area, backpressure is implicitly built up towards the source, which then stops sending out packets until the congestion is released. BMCC takes care that every receiving node will receive packets at the same rate. An alternative mode of operation, BMCC with Backpressure Pruning (BMCC-BP) allows the cutting of congested branches for single packets, permitting a higher rate for uncongested receivers.

Chapter 7 presented two different implementations of two of the above-mentioned multicast protocols. The first one (Section 7.1) is an implementation of SPBM for the Linux kernel that allows IP applications to send data via UDP to a group of receivers in an ad-hoc network. In the network

stack, the implementation resides on layer 2.5, between the MAC layer and the network/IP layer. It is compatible with unmodified standard kernels of versions 2.4 and 2.6, and may be compiled for x86 or ARM processor architectures.
The second implementation (Section 7.2) is an implementation of CBMF for the ScatterWeb MSB430 sensor nodes. Due to their low-level programmability they allow an integration of our routing protocol with the medium access control. The absence of periodic beacon messages makes the protocol especially suitable for energy-constrained sensor networks. During realization, a series of other constraints had to be respected: Limited memory and computational power demanded special consideration in order to avoid spoiling the correct operation of the protocol.

8.2 Outlook

Having extensively studied position-based multicast in design, simulation and small real-world setups, it remains to perform large-scale experiments and, finally, deployments, e. g., in metropolean-area wireless mesh networks [90]. As a preparation to the deployment, simulations employing different radio models which better meet the conditions of reality could provide valuable results.
With preset positions in static mesh networks the SPBM protocol is a good choice, since the frequency of SPBM beacon messages could be considerably decreased. At the same time, it would not be necessary to combine the routing protocol with a MAC layer, allowing deployment of the protocol on top of existing standard hardware. However, slight modifications of the MAC firmware would allow to deploy SPBM together with BMCC, which should be seen as an integral building block of position-based multicast since it perfectly optimizes the packet delivery performance for multicast applications.

APPENDIX A

Area of the Outer Part of a Reuleaux Forwarding Region

Corollary 2. *The area of the outer part of a Reuleaux triangle R with radius r given as*

$$O = \bigcup P(x_P|y_P) \mid P \in R \wedge \sqrt{x_P^2 + y_P^2} > \frac{r}{2} \tag{A.1}$$

covers approximately 68.8 percent of R.

Proof. Figure A.1 shows the Reuleaux triangle R as $\triangle ACE$. The outer part, described by Equation A.1, is limited by the points A, B, D, and E.

Let $2a$ be the radius of the Reuleaux triangle. Then the small circle around the origin with radius a is given as the equation

$$\begin{aligned} x^2 + y^2 &= a^2 \\ y &= \pm\sqrt{a^2 - x^2} \end{aligned}$$

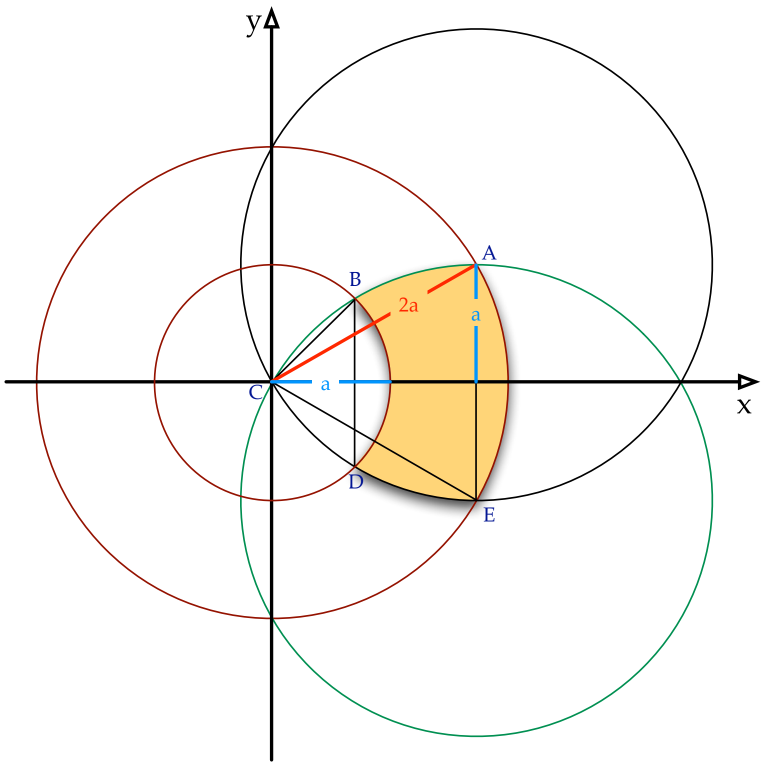


Figure A.1: *A Reuleaux forwarding area* and the circle around E is given as

$$\begin{aligned}\left(x-\sqrt{3}a\right)^2+(y+a)^2 &= 4a^2\\ y &= \pm\sqrt{-x^2+2\sqrt{3}ax+a^2}-a.\end{aligned}$$

The x-coordinate x_B of the intersection B of the two circles can be determined by setting

$$\begin{aligned}\sqrt{a^2-x_B^2} &= \sqrt{-x_B^2+2\sqrt{3}ax_B+a^2}-a\\ x_B &= a\left(\frac{\sqrt{15}+\sqrt{3}}{8}\right).\end{aligned}$$

Now we can calculate the area of the inner part of the Reuleaux triangle $I = R \setminus O$ as the sum of its part left of x_B

$$\begin{aligned}A_I^l &= 2\int_0^{a\frac{\sqrt{15}+\sqrt{3}}{8}} \sqrt{-x^2+2\sqrt{3}ax+a^2}-a\,dx\\ &= 8a^2\left(\arctan\frac{\sqrt{15}}{15}+\frac{7}{4}\sqrt{3}-\frac{15}{4}\sqrt{15}\right)\end{aligned}$$

and its part right of x_B

$$\begin{aligned}A_I^r &= 2\int_{a\cdot\frac{\sqrt{15}+\sqrt{3}}{8}}^{a} \sqrt{a^2-x^2}dx\\ &= a^2\left(\frac{2}{3}\arctan\frac{\sqrt{15}}{9}+\frac{1}{6}\pi-\frac{1}{32}\sqrt{15}-\frac{7}{32}\sqrt{3}\right).\end{aligned}$$

Together, this can be reduced to:

$$A_I = 2a^2\left(\arctan\frac{\sqrt{15}}{39}+\frac{5}{12}\pi-\frac{1}{4}\sqrt{15}\right).$$

In [8], the area of the Reuleaux triangle R is given as

$$A_R = 2a^2\left(\pi - \sqrt{3}\right).$$

Thus, the sought-after coverage is:

$$\begin{aligned}\frac{A_O}{A_R} &= 1 - \frac{A_I}{A_R} \\ &= 1 - \frac{\arctan\frac{\sqrt{15}}{39} + \frac{5}{12}\pi - \frac{1}{4}\sqrt{15}}{\pi - \sqrt{3}} \\ &\approx 0.688 \quad .\end{aligned}$$

□

Bibliography

[1] Apple Bonjour. `http://developer.apple.com/networking/bonjour/index.html`, 2007. [Online; last accessed 2007-07-10]. (Cited on page 6.)

[2] CoRe Group Implementation Portal. `http://core.it.uu.se/core/index.php/Main_Page`, 2007. [Online; last accessed 2007-09-24]. (Cited on page 134.)

[3] Kernel AODV from the National Institute of Standards an Technology (NIST). `http://w3.antd.nist.gov/wctg/aodv_kernel/`, 2004. [Online; last accessed 2007-09-24]. (Cited on page 134.)

[4] MeshDynamics. `http://www.meshdynamics.com/`, 2007. [Online; last accessed 2007-06-26]. (Cited on page 5.)

[5] The ns-2 network simulator. `http://nsnam.isi.edu/nsnam`, 2007. [Online; last accessed 2007-07-21]. (Cited on pages 42, 76, 97 and 120.)

[6] Wireless Multicast Extensions for ns-2.1b8. `http://www.monarch.cs.rice.edu/multicast_extensions.html`, 2006. [Online; last accessed 2006-04-26]. (Cited on pages 76 and 121.)

[7] INRIA HIPERCOM OLSR Web Page. `http://hipercom.inria.fr/olsr/`, 2007. [Online; last accessed 2007-09-24]. (Cited on page 134.)

[8] Wolfram MathWorld: The Reuleaux Triangle. `http://mathworld.wolfram.com/ReuleauxTriangle.html`, 2007. [Online; last accessed 2007-09-14]. (Cited on page 160.)

[9] MIT RoofNet Homepage. `http://pdos.csail.mit.edu/roofnet/doku.php`, 2007. [Online; last accessed 2007-06-26]. (Cited on page 5.)

[10] ScatterWeb GmbH. `http://www.scatterweb.de/`, 2007. [Online; last accessed 2007-09-26]. (Cited on pages 144 and 145.)

[11] SPBM kernel module web page, University of Mannheim. `http://www.informatik.uni-mannheim.de/pi4/projects/pbm/kernel.html`, 2006. [Online; last accessed 2006-04-26]. (Cited on pages 134 and 142.)

[12] Tropos Networks. `http://www.tropos.com/`, 2007. [Online; last accessed 2007-06-26]. (Cited on page 5.)

[13] Wi-Fi Alliance. `http://www.wi-fi.org/`, 2007. [Online; last accessed 2007-06-26]. (Cited on page 4.)

[14] ZigBee Alliance. `http://www.zigbee.org/`, 2007. [Online; last accessed 2007-09-26]. (Cited on page 144.)

[15] ABRAMSON, N. The ALOHA System—Another Alternative for Computer Communications. In *Proceedings of the AFIPS Fall Joint Computer Conference (FJCC '70)* (Houston, TX, USA, November 1970), vol. 37, pp. 281–285. (Cited on page 2.)

[16] ABRAMSON, N. Development of the ALOHANET. *IEEE Transactions on Information Theory 31*, 2 (March 1985), 119–123. (Cited on page 2.)

[17] ADAMS, A., NICHOLAS, J., AND SIADAK, W. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). RFC 3973, `http://www.ietf.org/rfc/rfc3973.txt`, 2005. Status: EXPERIMENTAL. (Cited on page 6.)

[18] AGUILAR, L. Datagram Routing for Internet Multicasting. In *Proceedings of the ACM Symposium on Communications architectures and protocols (SIGCOMM '84)* (Montréal, Quebec, Canada, June 1984), pp. 58–63. (Cited on page 5.)

[19] TAPR Standard AX.25, Version 2.2. Tucson Amateur Packet Radio Corporation, July 1998. (Cited on page 2.)

[20] BASAGNI, S., CHLAMTAC, I., AND SYROTIUK, V. R. Location aware, dependable multicast for mobile ad hoc networks. *Computer Networks 36*, 5–6 (August 2001), 659–670. (Cited on pages 30, 31 and 35.)

[21] BASAGNI, S., CHLAMTAC, I., SYROTIUK, V. R., AND TALEBI, R. On-Demand Location Aware Multicast (OLAM) for Ad Hoc Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2000)* (Chicago, IL, USA, September 2000), vol. 3, pp. 1323–1328. (Cited on pages 30 and 31.)

[22] BASAGNI, S., CHLAMTAC, I., SYROTIUK, V. R., AND WOODWARD, B. A. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)* (Dallas, TX, USA, October 1998), pp. 76–84. (Cited on pages 34 and 36.)

[23] BAUMUNG, P. Stable, Congestion-Controlled Application-Layer Multicasting in Pedestrian Ad-hoc Networks. In *Proceedings of the Sixth IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2005)* (Taormina - Giardini Naxos, Italy, June 2005), pp. 57–64. (Cited on page 108.)

[24] BAUMUNG, P., ZITTERBART, M., AND KUTZNER, K. Improving Delivery Ratios for Application Layer Multicast in Mobile Ad-hoc Networks. In *Proc. of the 4th Workshop on Applications and Services in Wireless Networks (ASWN 2004)* (Boston, MA, USA, August 2004), pp. 132–141. (Cited on page 108.)

[25] BAUMUNG, P., ZITTERBART, M., AND KUTZNER, K. Improving Delivery Ratios for Application Layer Multicast in Mobile Ad-hoc Networks. *Elsevier Computer Communications 18*, 14 (September 2005), 1669–1679. (Cited on page 108.)

[26] BETTSTETTER, C. Mobility Modeling in Wireless Networks: Categorization, Smooth Movement, and Border Effects. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R) 5*, 3 (July 2001), 55–67. (Cited on pages 45, 77 and 98.)

[27] BETTSTETTER, C., HARTENSTEIN, H., AND PÉREZ-COSTA, X. Stochastic Properties of the Random Waypoint Mobility Model. *ACM/Kluwer Wireless Networks, Special Issue on Modeling & Analysis of Mobile Networks 9*, 2 (2003). (Cited on pages 45, 77 and 98.)

[28] BICKET, J., AGUAYO, D., BISWAS, S., AND MORRIS, R. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *Proceedings of the Eleventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2005)* (Cologne, Germany, August/September 2005), pp. 31–42. (Cited on page 5.)

[29] BIERSACK, E. W. Where is Multicast Today? *ACM SIGCOMM Computer Communication Review (CCR) 35*, 5 (October 2005), 83–84. (Cited on page 6.)

[30] BISWAS, S. Opportunistic Routing in Multi-Hop Wireless Networks. Master's thesis, M.I.T., March 2005. (Cited on page 87.)

[31] BISWAS, S., AND MORRIS, R. Opportunistic Routing in Multi-Hop Wireless Networks. *ACM SIGCOMM Computer Communication Review (CCR) 34*, 1 (January 2004), 69–74. (Cited on page 87.)

[32] BISWAS, S., AND MORRIS, R. ExOR: Opportunistic Routing in Multi-Hop Wireless Networks. In *Proceedings of the 2005 Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2005)* (Philadelphia, PA, USA, August 2005), pp. 133–144. (Cited on page 87.)

[33] BLOOM, B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM 13*, 7 (July 1970), 422–426. (Cited on page 59.)

[34] BLUM, B., HE, T., SON, S., AND STANKOVIC, J. IGF: A State-Free Robust Communication Protocol for Wireless Sensor Networks. Tech. Rep. CS-2003-11, Department of Computer Science, University of Virginia, 2003. (Cited on page 87.)

[35] BOGGS, D. R. *Internet Broadcasting*. PhD thesis, Stanford University, July 1982. (Cited on page 5.)

[36] BOSE, P., MORIN, P., STOJMENOVIC, I., AND URRUTIA, J. Routing with guaranteed delivery in ad hoc Wireless Networks. In *Proceedings of the 3rd international Workshop on Discrete Algorithms and Methods for Mobile Computing and communications (DIAL-M 1999)* (Seattle, WS, USA, August 1999), pp. 48–55. (Cited on pages 12, 33, 40, 56 and 74.)

[37] BRIESEMEISTER, L., AND HOMMEL, G. Role-Based Multicast in Highly Mobile but Sparsely Connected Ad Hoc Networks. In *Proceedings of the First ACM international Symposium on Mobile and Ad Hoc Networking & Computing (MobiHoc 2000)* (Boston, MA, USA, August 2000), pp. 45–50. (Cited on page 30.)

[38] BROCH, J., MALTZ, D. A., JOHNSON, D. B., HU, Y.-C., AND JETCHEVA, J. G. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)* (Dallas, TX, USA, October 1998), pp. 85–97. (Cited on pages 44 and 77.)

[39] BUTTER, T. Contention-Based Multicast Forwarding for Mobile Ad-Hoc Networks. Master's thesis, Department of Mathematics and Computer Science, University of Mannheim, 2004. (Cited on pages 69 and 85.)

[40] CAMP, T., BOLENG, J., AND DAVIES, V. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications 2*, 5 (2002), 483–502. (Cited on pages 45, 77 and 98.)

[41] CAPKUN, S., HAMDI, M., AND HUBAUX, J.-P. GPS-free positioning in mobile ad hoc networks. *Cluster Computing Journal 5*, 2 (2002). (Cited on page 10.)

[42] CHAKERES, I. D., AND BELDING-ROYER, E. M. AODV Routing Protocol Implementation Design. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCS Workshops)* (Hachioji, Tokyo, Japan, March 2004), pp. 698–703. (Cited on page 134.)

[43] CHEN, K., AND NAHRSTEDT, K. Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)* (New York City, NY, USA, June 2002), pp. 1192–1201. (Cited on pages 30, 31 and 35.)

[44] CHERITON, D. R., AND DEERING, S. E. Host Groups: a Multicast Extension for Datagram Internetworks. In *Proceedings of the Ninth Symposium on Data Communications (SIGCOMM '85)* (Whistler Mountain,

British Columbia, Canada, September 1985), pp. 172–179. (Cited on page 5.)

[45] CHESHIRE, S., AND KROCHMAL, M. Multicast DNS. Internet Draft, draft-cheshire-dnsext-multicastdns-06.txt, work in progress, August 2006. (Cited on page 6.)

[46] CHESHIRE, S., AND STEINBERG, D. *Zero Configuration Networking: The Definitive Guide*. O'Reilly Media, Inc., 2005. (Cited on page 6.)

[47] CHIANG, C.-C., GERLA, M., AND ZHANG, L. Shared Tree Wireless Network Multicast. In *Proceedings of the 6th IEEE International Conference on Computer Communications and Networks (ICCCN 1997)* (Las Vegas, NV, USA, September 1997), pp. 28–33. (Cited on pages 22 and 30.)

[48] CHIANG, C.-C., GERLA, M., AND ZHANG, L. Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks. *ACM-Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing 1*, 2 (December 1998), 187–196. (Cited on pages 25 and 30.)

[49] CLAUSEN, T., AND JACQUET, P. Optimized Link State Routing Protocol (OLSR). RFC 3626, `http://www.ietf.org/rfc/rfc3626.txt`, 2003. Status: EXPERIMENTAL. (Cited on page 4.)

[50] CORSON, M. S., AND BATSELL, S. G. A Reservation-Based Multicast (RBM) Routing Protocol for Mobile Networks: Initial Route Construction Phase. *Wireless Networks 1*, 4 (1995), 427–450. (Cited on pages 9, 14 and 30.)

[51] CORSON, M. S., AND BATSELL, S. G. A Reservation-Based Multicast (RBM) Routing Protocol for Mobile Networks: Overview of Initial Route Construction. In *Proceedings of the 14th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1995)* (Boston, MA, USA, April 1995), pp. 1063–1074. (Cited on page 30.)

[52] CORSON, M. S., AND EPHREMIDES, A. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks 1*, 1 (February 1995), 61–81. (Cited on page 30.)

[53] CORSON, S., AND MACKER, J. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501, `http://www.ietf.org/rfc/rfc2501.txt`, 1999. Status: INFORMATIONAL. (Cited on page 4.)

[54] CROWCROFT, J., AND PHILLIPS, I. *TCP/IP and Linux Protocol Implementation*. John Wiley & Sons, 2002. (Cited on page 137.)

[55] DAS, S. K., MANOJ, B. S., AND MURTHY, C. S. R. A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks. In *Proceedings of the Third ACM international Symposium on Mobile and Ad Hoc Networking & Computing (MobiHoc 2002)* (Lausanne, Switzerland, June 2002), pp. 24–35. (Cited on pages 27 and 30.)

[56] DE MORAIS CORDEIRO, C., GOSSAIN, H., AND AGRAWAL, D. P. Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions. *IEEE Network 17*, 1 (2003). (Cited on page 22.)

[57] DEERING, S., AND CHERITON, D. Host groups: A multicast extension to the Internet Protocol. RFC 966, `http://www.ietf.org/rfc/rfc966.txt`, 1985. Status: UNKNOWN. (Cited on page 5.)

[58] DEERING, S. E. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of the Symposium on Communications architectures and protocols (SIGCOMM '88)* (Stanford, CA, USA, August 1988), pp. 55–64. (Cited on page 5.)

[59] DEERING, S. E. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, December 1991. (Cited on page 5.)

[60] DEERING, S. E., AND CHERITON, D. R. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems (ToCS) 8*, 2 (May 1990). (Cited on page 5.)

[61] DEERING, S. E., ESTRIN, D., FARINACCI, D., JACOBSON, V., LIU, C.-G., AND WEI, L. The PIM Architecture for Wide-Area Multicast Routing. *IEEE/ACM Transactions on Networking (ToN) 4*, 2 (April 1996), 153–162. (Cited on pages 14 and 22.)

[62] DEVARAPALLI, V., AND SIDHU, D. MZR: A Multicast Protocol for Mobile Ad Hoc Networks. In *Proceedings of the IEEE International Conference on Communications (ICC 2001)* (Helsinki, Finland, June 2001), vol. 3, pp. 886–891. (Cited on page 30.)

[63] ERIKKSON, H. MBONE: the Multicast Backbone. *Communications of the ACM 37*, 8 (August 1994), 54–60. (Cited on page 5.)

[64] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and Wei, L. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2117, http://www.ietf.org/rfc/rfc2117.txt, 1997. Status: EXPERIMENTAL. (Cited on page 6.)

[65] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and Wei, L. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2362, http://www.ietf.org/rfc/rfc2362.txt, 1998. Status: EXPERIMENTAL. (Cited on page 14.)

[66] Fenner, B., Handley, M., Holbrook, H., and Kouvelas, I. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601, http://www.ietf.org/rfc/rfc4601.txt, 2006. Status: PROPOSED STANDARD. (Cited on pages 6 and 16.)

[67] Füssler, H. *Position-Based Packet Forwarding for Vehicular Ad-Hoc Networks*. PhD thesis, University of Mannheim, April 2007. (Cited on pages 12, 34 and 88.)

[68] Füssler, H., Leibscher, A., Widmer, J., Transier, M., and Effelsberg, W. Contention-Based Distance-Vector Routing (CBDV) for Mobile Ad-Hoc Networks. In *12th IEEE International Conference on Networking Protocols—Student Poster Session(ICNP 2004)* (Berlin, Germany, October 2004). (Cited on page 12.)

[69] Füssler, H., Möske, M., Hartenstein, H., Franz, W., Festag, A., and Wagner, C. A Position-Based Router: Design, Implementation and Measurements. In *Inter-Vehicle-Communications Based on Ad Hoc Networking Principles—The FleetNet Project*, W. Franz, H. Hartenstein, and M. Mauve, Eds. Universitätsverlag Karlsruhe, Karlsruhe, Germany, November 2005, pp. 145–174. (Cited on page 134.)

[70] Füssler, H., Widmer, J., Käsemann, M., Mauve, M., and Hartenstein, H. Beaconless Position-Based Routing for Mobile Ad-Hoc Networks. Tech. Rep. TR-03-001, Department of Computer Science, University of Mannheim, 2003. (Cited on page 87.)

[71] FÜSSLER, H., WIDMER, J., KÄSEMANN, M., MAUVE, M., AND HARTENSTEIN, H. Contention-Based Forwarding for Mobile Ad-Hoc Networks. *Elsevier's Ad Hoc Networks 1*, 4 (2003), 351–369. (Cited on pages 82, 86, 87 and 95.)

[72] FÜSSLER, H., WIDMER, J., MAUVE, M., AND HARTENSTEIN, H. A Novel Forwarding Paradigm for Position-Based Routing (with Implicit Addressing). In *Proceedings of the IEEE 18th Annual Workshop on Computer Communications (CCW 2003)* (Dana Point, CA, USA, October 2003), pp. 194–200. (Cited on page 87.)

[73] GABRIEL, K., AND SOKAL, R. A new statistical approach to geographic variation analysis. *Systematic Zoology 18* (1969), 259–278. (Cited on page 40.)

[74] GARCIA-LUNA-ACEVES, J. J., AND MADRUGA, E. L. A Multicast Routing Protocol for Ad-Hoc Networks. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1999)* (New York City, NY, USA, March 1999), pp. 784–792. (Cited on pages 28 and 30.)

[75] GIORDANO, S., AND HAMDI, M. Mobility Management: The Virtual Home Region. Tech. Rep. SSC/1999/037, EPFL-ICA, October 1999. (Cited on page 34.)

[76] GIORDANO, S., STOJMENOVIC, I., AND BLAŽEVIĆ, L. Position-Based Routing Algorithms for AdHoc Networks: A Taxonomy. *Ad Hoc Wireless Networking* (November 2003). (Cited on page 12.)

[77] GOSSAIN, H., NANDIRAJU, N., ANAND, K., AND AGRAWAL, D. P. Supporting MAC Layer Multicast in IEEE 802.11 based MANETs: Issues and Solutions. In *Proceedings of the 29th Annual IEEE Conference on Local Computer Networks (LCN 2004)* (Tampa, FL, USA, November 2004), pp. 172–179. (Cited on page 108.)

[78] GUPTA, S. K. S., AND SRIMANI, P. K. An Adaptive Protocol for Reliable Multicast in Mobile Multi-hop Radio Networks. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999)* (New Orleans, LA, USA, February 1999), pp. 111–122. (Cited on page 30.)

[79] HARTENSTEIN, H., FÜSSLER, H., MAUVE, M., AND FRANZ, W. Simulation Results and Proof-of-Concept Implementation of the FleetNet

Position-Based Router. In *Proceedings of the IFIP-TC6 8th International Conference on Personal Wireless Communications (PWC 2003)* (Venice, Italy, September 2003), pp. 192–197. (Cited on page 134.)

[80] HASSAN, J., AND JHA, S. On the Optimization Trade-Offs of Expanding Ring Search. *Lecture Notes in Computer Science 3326* (January 2004), 489–494. (Cited on page 20.)

[81] HEISSENBÜTTEL, M., AND BRAUN, T. A Novel Position-based and Beacon-less Routing Algorithm for Mobile Ad-Hoc Networks. In *Proc. of the 3rd Workshop on Applications and Services in Wireless Networks (ASWN 2003)* (Bern, Switzerland, July 2003), pp. 197–209. (Cited on page 87.)

[82] HEISSENBÜTTEL, M., AND BRAUN, T. BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks. *Elsevier Computer Communications 27*, 11 (2004), 1076–1086. (Cited on page 87.)

[83] HEISSENBÜTTEL, M., BRAUN, T., AND ROTH, T. GNU/Linux Implementation of a Position-based Routing Protocol. In *Proceedings of IEEE ICPS Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality (REALMAN 2005)* (Santorini, Greece, July 2005), pp. 27–35. (Cited on page 134.)

[84] HIGHTOWER, J., AND BORRIELLO, G. A Survey and Taxonomy of Location Systems for Ubiquitous Computing. Tech. Rep. 01-08-03, University of Washington, Department of Computer Science and Engineering, August 2001. (Cited on pages 10 and 36.)

[85] HOLBROOK, H. W., SINGHAL, S. K., AND CHERITON, D. R. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM 1995)* (Cambridge, MA, USA, August 1995), pp. 328–341. (Cited on page 113.)

[86] HWANG, F. K.-M., RICHARDS, D. S., AND WINTER, P. The Steiner Tree Problem. vol. 53 of *Annals of Discrete Mathematics*. Elsevier Scientific Publishers, Amsterdam, The Netherlands, 1992, pp. 203–282. (Cited on pages 31 and 35.)

[87] ANSI/IEEE Standard 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. LAN/MAN

Standards Committee of the IEEE Computer Society, 1999. (Cited on pages 4, 19, 96, 133, 144 and 147.)

[88] IEEE Standard 802.11b: Higher-Speed Physical Layer Extension in the 2.4 GHz Band. Supplement to ANSI/IEEE Std. 802.11, 1999 Edition, LAN/MAN Standards Committee of the IEEE Computer Society, 1999. (Cited on page 4.)

[89] IEEE Standard 802.11g: Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band. Amendment to IEEE Std. 802.11, 1999 Edition, LAN/MAN Standards Committee of the IEEE Computer Society, 2003. (Cited on page 4.)

[90] IEEE Draft 802.11s: Amendment: Wireless Local Area Mesh Networking. Draft for an Amendment to IEEE Std. 802.11, 1999 Edition, LAN/MAN Standards Committee of the IEEE Computer Society, April 2007. (Cited on pages 5 and 156.)

[91] IEEE Standard 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). LAN/MAN Standards Committee of the IEEE Computer Society, September 2006. (Cited on page 144.)

[92] JAIN, R. K., CHIU, D.-M. W., AND HAWE, W. R. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. Tech. Rep. DEC-TR-301, Digital Equipment Corporation, Eastern Research Lab, September 1984. (Cited on page 126.)

[93] JAIN, S., AND DAS, S. R. MAC Layer Multicast in Wireless Multihop Networks. In *Proceedings of the First International Conference on Communication System Software and Middleware (Comsware 2006)* (Delhi, India, January 2006). (Cited on page 108.)

[94] JETCHEVA, J. G., HU, Y.-C., MALTZ, D. A., AND JOHNSON, D. B. A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks. Internet Draft, draft-ietf-manet-simple-mbcast-01.txt, expired January 2002, July 2001. (Cited on pages 4 and 30.)

[95] JETCHEVA, J. G., AND JOHNSON, D. B. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the Second ACM international Symposium on Mobile and Ad*

Hoc Networking & Computing (MobiHoc 2001) (Long Beach, CA, USA, October 2001), pp. 33–44. (Cited on page 30.)

[96] JETCHEVA, J. G., AND JOHNSON, D. B. The Adaptive Demand-Driven Multicast Routing Protocol for Mobile Ad Hoc Networks (ADMR). Internet Draft, draft-ietf-manet-admr-00.txt, expired January 2002, July 2001. (Cited on page 4.)

[97] JI, L., AND CORSON, M. S. A Lightweight Adaptive Multicast Algorithm. In *Proceedings of IEEE Global Telecommunications Conference (Globecom 1998)* (Sydney, Australia, November 1998), pp. 1036–1042. (Cited on page 30.)

[98] JI, L., AND CORSON, M. S. Differential Destination Multicast (DDM) Specification. Internet Draft, draft-ietf-manet-ddm-00.txt, expired January 2001, July 2000. (Cited on page 4.)

[99] JI, L., AND CORSON, M. S. Differential Destination Multicast—A MANET Multicast Routing Protocol for Small Groups. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)* (Anchorage, AK, USA, April 2001), pp. 1192–1202. (Cited on page 30.)

[100] JOHNSON, D., HU, Y., AND MALTZ, D. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, `http://www.ietf.org/rfc/rfc4728.txt`, 2007. Status: EXPERIMENTAL. (Cited on page 4.)

[101] JOHNSON, D. B., AND MALTZ, D. A. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, T. Imielinski and H. Korth, Eds., vol. 353. Kluwer Academic Publishers, 1996, pp. 153–181. (Cited on pages 30, 31, 44 and 77.)

[102] JUBIN, J., AND TORNOW, J. D. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE 75*, 1 (January 1987), 21–32. (Cited on pages 3 and 109.)

[103] KAHN, R. E. The Organization of Computer Resources into a Packet Radio Network. *IEEE Transactions on Communications 25*, 1 (January 1977), 169–178. (Cited on page 3.)

[104] KAHN, R. E., GRONEMEYER, S. A., BURCHFIEL, J., AND KUNZELMAN, R. C. Advances in Packet Radio Technology. *Proceedings of the IEEE 66*, 11 (November 1978), 1468–1496. (Cited on page 3.)

[105] KAPLAN, E. B. *Understanding GPS*. Artech House, 1996. (Cited on pages 10 and 36.)

[106] KARP, B. N., AND KUNG, H. T. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)* (Boston, MA, USA, August 2000), pp. 243–254. (Cited on pages 12, 33, 40, 56 and 74.)

[107] KIESS, W., AND MAUVE, M. A Survey on Real-World Implementations of Mobile Ad-hoc Networks. *Elsevier's Ad Hoc Networks 5*, 3 (April 2007), 324–339. (Cited on page 134.)

[108] KING, T., BUTTER, T., LEMELSON, H., HAENSELMANN, T., AND EFFELSBERG, W. Loclib,trace,eva,ana: Research Tools for 802.11-based Positioning Systems. In *Proceedings of the 2nd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WinTECH 2007)* (Montreal, Quebec, Canada, September 2007), pp. 67–74. (Cited on page 142.)

[109] KING, T., HAENSELMANN, T., KOPF, S., AND EFFELSBERG, W. Positionierung mit Wireless-LAN und Bluetooth. *PIK - Praxis der Informationsverarbeitung und Kommunikation 29*, 1 (March 2006), 9–17. (Cited on pages 10 and 36.)

[110] KIRSCH, R. Implementation of a Distance-Vector-Based Recovery-Strategy for Position-Based-Routing, January 2007. (Cited on page 12.)

[111] KLEINROCK, L., AND TOBAGI, F. A. Packet Switching in Radio Channels: Part I – Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics. *IEEE Transactions on Communications 23*, 12 (December 1975), 1400–1416. (Cited on page 3.)

[112] KO, Y.-B., AND VAIDYA, N. H. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)* (Dallas, TX, USA, October 1998), pp. 66–75. (Cited on page 30.)

[113] KO, Y.-B., AND VAIDYA, N. H. Location-Based Multicast in Mobile Ad Hoc Networks. Tech. Rep. 98-018, Texas A&M University, 1998. (Cited on page 30.)

[114] KO, Y.-B., AND VAIDYA, N. H. Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999)* (New Orleans, LA, USA, February 1999), pp. 101–110. (Cited on page 30.)

[115] KUNZ, T., AND CHENG, E. Multicasting in Ad-Hoc Networks: Comparing MAODV and ODMRP. In *Proceedings of the 7th European Conference on Computer Supported Cooperative Work (ECSCW 2001)* (Bonn, Germany, September 2001). (Cited on page 22.)

[116] KÄSEMANN, M. Beaconless Position-Based Routing for Mobile Ad-Hoc Networks. Master's thesis, Department of Mathematics and Computer Science, University of Mannheim, February 2003. (Cited on pages 87 and 88.)

[117] KÄSEMANN, M., FÜSSLER, H., HARTENSTEIN, H., AND MAUVE, M. A Reactive Location Service for Mobile Ad Hoc Networks. Tech. Rep. TR-02-014, Department of Computer Science, University of Mannheim, November 2002. (Cited on page 11.)

[118] KÄSTLE, J. Contention-Based Forwarding for Uni- and Multicast in Sensor Networks. Master's thesis, Department of Mathematics and Computer Science, University of Mannheim, September 2005. (Cited on pages 94 and 133.)

[119] LANG, T. Location Based Multicast for Wireless Networks. Master's thesis, Department of Computer Science, University of Mannheim, 2002. (Cited on page 33.)

[120] LEE, M., AND KIM, Y. K. PatchODMRP: An Ad-hoc Multicast Routing Protocol. In *Proceedings of the 15th IEEE International Conference on Information Networking* (Beppu City, Oita, Japan, February 2001), pp. 537–543. (Cited on pages 26 and 30.)

[121] LEE, S., AND KIM, C. Neighbor Supporting Ad hoc Multicast Routing Protocol. In *Proceedings of the First ACM international Symposium on Mobile and Ad Hoc Networking & Computing (MobiHoc 2000)* (Boston, MA, USA, August 2000), pp. 37–44. (Cited on pages 27 and 30.)

[122] LEE, S.-J., GERLA, M., AND CHIANG, C.-C. On-Demand Multicast Routing Protocol. In *Proceedings of IEEE Wireless Communications*

and Networking Conference (WCNC 1999) (New Orleans, LA, USA, September 1999), pp. 1298–1302. (Cited on pages 25, 30, 107 and 121.)

[123] LEE, S.-J., SU, W., AND GERLA, M. On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks. Internet Draft, draft-ietf-manet-odmrp-04.txt, expired February 2003, November 2002. (Cited on pages 4 and 77.)

[124] LEE, S.-J., SU, W., HSU, J., GERLA, M., AND BAGRODIA, R. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)* (Tel Aviv, Israel, March 2000), pp. 565–574. (Cited on page 22.)

[125] LEIBSCHER, A. Recovery-Strategies for Beaconless, Position-Based Routing in Mobile Ad-hoc Networks. Master's thesis, Department of Mathematics and Computer Science, University of Mannheim, March 2004. (Cited on page 12.)

[126] LI, J., JANNOTTI, J., DECOUTO, D. S. J., KARGER, D. R., AND MORRIS, R. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)* (Boston, MA, USA, August 2000), pp. 120–130. (Cited on page 34.)

[127] LIN, C. R., AND CHAO, S.-W. A Multicast Routing Protocol for Multihop Wireless Networks. In *Proceedings of IEEE Global Telecommunications Conference (Globecom 1999)* (Rio de Janeiro, Brazil, December 1999), pp. 235–239. (Cited on page 30.)

[128] LIU, M., TALPADE, R. R., MCAULEY, A., AND BOMMAIAH, E. AMRoute: Adhoc Multicast Routing Protocol. Tech. Rep. TR 99-1, CSHCN, University of Maryland, 1999. (Cited on pages 23 and 30.)

[129] LUNDGREN, H., LUNDBERG, D., NIELSEN, J., NORDSTRÖM, E., AND TSCHUDIN, C. A large-scale testbed for reproducible Ad Hoc protocol evaluations. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2002)* (Orlando, FL, USA, March 2002), pp. 337–343. (Cited on page 134.)

[130] MACKER, J. Simple Multicast Forwarding for MANET. Internet Draft, draft-ietf-manet-smf-05.txt, work in progress, June 2007. (Cited on page 4.)

[131] MACKER, J., AND CORSON, M. S. Mobile ad hoc networking and the IETF. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R) 2*, 1 (January 1998), 9–14. (Cited on page 4.)

[132] MAUVE, M., FÜSSLER, H., WIDMER, J., AND LANG, T. Position-Based Multicast Routing for Mobile Ad-Hoc Networks. Tech. Rep. TR-03-004, Department of Computer Science, University of Mannheim, 2003. (Cited on page 33.)

[133] MAUVE, M., FÜSSLER, H., WIDMER, J., AND LANG, T. Position-based multicast routing for mobile Ad-hoc networks. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R) 7*, 3 (July 2003), 53–55. (Cited on page 33.)

[134] MAUVE, M., FÜSSLER, H., WIDMER, J., AND LANG, T. Poster: Position-Based Multicast Routing for Mobile Ad-Hoc Networks. In *Proceedings of the Fourth ACM international Symposium on Mobile and Ad Hoc Networking & Computing (MobiHoc 2003, electronic edition)* (Annapolis, MD, USA, June 2003). (Cited on page 33.)

[135] MAUVE, M., WIDMER, J., AND HARTENSTEIN, H. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network 15*, 6 (November/December 2001), 30–39. (Cited on pages 12, 73, 88 and 134.)

[136] MEGGERS, J., AND FILIOS, G. Multicast Communication in ”ad hoc” Networks. In *Proceedings of the IEEE 48th Vehicular Technology Conference (VTC 1998)* (Ottawa, Canada, May 1998), pp. 372–376. (Cited on page 30.)

[137] MORRIS, R., JANNOTTI, J., KAASHOEK, F., LI, J., AND DECOUTO, D. S. J. CarNet: A Scalable Ad Hoc Wireless Network System. In *Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System* (Kolding, Denmark, September 2000). (Cited on page 34.)

[138] MOUSTAFA, H., AND LABIOD, H. SRMP: A Mesh-based Protocol for Multicast Communication in ad hoc networks. In *2002 International Conference on Third Generation Wireless and Beyond* (San Francisco, CA, May 2002), pp. 43–48. (Cited on page 30.)

[139] MOUSTAFA, H., AND LABIOD, H. A Performance Comparison of Multicast Routing Protocols in Ad hoc Networks. In *Proceedings of the*

14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2003) (Beijing, China, September 2003), vol. 1, pp. 497–501. (Cited on page 22.)

[140] MOUSTAFA, H., AND LABIOD, H. A Performance Analysis of Source Routing-based Multicast Protocol (SRMP) Using Different Mobility Models. In *Proceedings of the IEEE International Conference on Communications (ICC 2004)* (Paris, France, June 2004), vol. 7, pp. 4192–4196. (Cited on page 30.)

[141] MOY, J. Multicast Extensions to OSPF. RFC 1584, `http://www.ietf.org/rfc/rfc1584.txt`, 1994. Status: PROPOSED STANDARD. (Cited on page 6.)

[142] NI, S.-Y., YU-CHEE, T., YUH-SHYAN, C., AND JANG-PING, S. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1999)* (Seattle, WA, USA, August 1999), pp. 151–162. (Cited on pages 11 and 19.)

[143] NONNENMACHER, J., AND BIERSACK, E. W. Scalable Feedback for Large Groups. *IEEE/ACM Transactions on Networking (ToN) 7*, 3 (1999), 375–386. (Cited on page 62.)

[144] OBRACZKA, K., AND TSUDIK, G. Multicast Routing Issues in Ad Hoc Networks. In *IEEE International Conference on Universal Personal Communications (ICUPC 1998)* (Florence, Italy, October 1998). (Cited on page 30.)

[145] OGIER, R., TEMPLIN, F., AND LEWIS, M. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684, `http://www.ietf.org/rfc/rfc3684.txt`, 2004. Status: EXPERIMENTAL. (Cited on page 4.)

[146] OZAKI, T., KIM, J. B., AND SUDA, T. Bandwidth-Efficient Multicast Routing Protocol for Ad-Hoc Networks. In *Proceedings of the 8th IEEE International Conference on Computer Communications and Networks (IC-CCN 1999)* (Boston, MA, USA, October 1999), pp. 1182–1191. (Cited on page 30.)

[147] OZAKI, T., KIM, J. B., AND SUDA, T. Bandwidth-Efficient Multicast Routing for Multihop, Ad-Hoc Wireless Networks. In *Proceedings of*

the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001) (Anchorage, AK, USA, April 2001), pp. 1182–1191. (Cited on page 30.)

[148] PARK, V. D., AND CORSON, M. S. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1997)* (Kobe, Japan, April 1997), pp. 1405–1413. (Cited on page 30.)

[149] PAWLIKOWSKI, K., JEONG, H.-D. J., AND LEE, J.-S. R. On Credibility of Simulation Studies of Telecommunications Networks. *IEEE Communications Magazine 40*, 1 (January 2002), 132–139. (Cited on page 134.)

[150] PENG, J., AND SIKDAR, B. A Multicast Congestion Control Scheme for Mobile Ad-hoc Networks. In *Proceedings of IEEE Global Telecommunications Conference (Globecom 2003)* (San Francisco, CA, USA, December 2003), vol. 5, pp. 2860–2864. (Cited on page 108.)

[151] PERKINS, C., BELDING-ROYER, E., AND DAS, S. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, `http://www.ietf.org/rfc/rfc3561.txt`, 2003. Status: EXPERIMENTAL. (Cited on pages 4 and 23.)

[152] PERKINS, C. E., AND ROYER, E. M. Ad-Hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999)* (New Orleans, LA, USA, February 1999), pp. 90–100. (Cited on pages 23 and 30.)

[153] PRÜFER, H. Neuer Beweis eines Satzes über Permutationen. *Archiv der Mathematik und Physik 3*, 27 (1918), 142–144. (Cited on page 31.)

[154] RADUNOVIĆ, B., AND BOUDEC, J.-Y. L. A Unified Framework for Max-Min and Min-Max Fairness with Applications. In *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2002)* (Monticello, IL, USA, October 2002). (Cited on page 117.)

[155] RAJENDRAN, V., OBRACZKA, K., YI, Y., LEE, S.-J., TANG, K., AND GERLA, M. Combining Source- and Localized Recovery to Achieve Reliable Multicast in Multi-Hop Ad Hoc Networks. In *Proceedings of*

the Third IFIP-TC6 Networking Conference (Networking 2004) (Athens, Greece, May 2004), pp. 112–124. (Cited on page 107.)

[156] REULEAUX, F. *Theoretische Kinematik. Grundzüge einer Theorie des Maschinenwesens*. F. Vieweg und Sohn, Braunschweig, Germany, 1875. (Cited on page 89.)

[157] REULEAUX, F. *The Kinematics of Machinery. Outlines of a Theory of Machines*. Macmillan and Co., London, UK, 1876. (Cited on page 89.)

[158] ROBERTS, L. G. Extension of Packet Communication Technology to a Hand-Held Personal Terminal. In *Proceedings of the Spring Joint Computer Conference* (Atlantic City, NJ, USA, May 1972), pp. 295–298. (Cited on page 2.)

[159] ROBERTS, L. G. ALOHA packet system with and without slots and capture. *ACM SIGCOMM Computer Communication Review (CCR) 5*, 2 (April 1975), 28–42. (Cited on page 2.)

[160] ROYER, E. M., MELLIAR-SMITH, P. M., AND MOSER, L. E. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proceedings of the IEEE International Conference on Communications (ICC 2001)* (Helsinki, Finland, June 2001), vol. 3, pp. 857–861. (Cited on pages 98 and 123.)

[161] ROYER, E. M., AND PERKINS, C. E. Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1999)* (Seattle, WA, USA, August 1999), pp. 207–218. (Cited on pages 23 and 30.)

[162] ROYER, E. M., AND PERKINS, C. E. Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing. Internet Draft, draft-ietf-manet-maodv-00.txt, expired December 2000, July 2000. (Cited on page 4.)

[163] ROYER, E. M., AND TOH, C.-K. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications* (April 1999), 46–55. (Cited on page 12.)

[164] SCHEUERMANN, B., LOCHERT, C., AND MAUVE, M. Implicit Hop-by-Hop Congestion Control in Wireless Multihop Networks. *Elsevier's Ad Hoc Networks* (2007). (Cited on pages 106 and 109.)

[165] SCHEUERMANN, B., TRANSIER, M., LOCHERT, C., MAUVE, M., AND EFFELSBERG, W. Backpressure Multicast Congestion Control in Mobile Ad-hoc Networks. In *Proceedings of the 3rd ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2007)* (New York City, NY, USA, December 2007). (Cited on page 105.)

[166] SHACHAM, N., AND WESTCOTT, J. Future directions in packet radio architectures and protocols. *Proceedings of the IEEE 75*, 1 (Januar 1987), 83–99. (Cited on page 4.)

[167] SINGH, H. Survey of Ad-hoc Multicast Protocols. available from `http://www.cs.pdx.edu/~harkirat/projects.html` [last accessed 2007-10-03], 2000. (Cited on page 22.)

[168] SINHA, P., SIVAKUMAR, R., AND BHARGHAVAN, V. MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 1999)* (New Orleans, LA, USA, September 1999), pp. 1313–1317. (Cited on pages 29 and 30.)

[169] SIVAKUMAR, R., SINHA, P., AND BHARGHAVAN, V. CEDAR: a Core-Extraction Distributed Ad Hoc Routing Algorithm. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1999)* (New York City, NY, USA, March 1999), pp. 202–209. (Cited on pages 29 and 30.)

[170] STANZE, O., AND ZITTERBART, M. Multicast-Routing in mobilen Ad-Hoc-Netzen. *Praxis der Informationsverarbeitung und Kommunikation 4* (December 2003). (Cited on page 22.)

[171] STEVENS, W. R., FENNER, B., AND RUDOFF, A. M. *UNIX Network Programming*, 3rd ed., vol. 1. Addison-Wesley, 2004. (Cited on page 139.)

[172] STOJMENOVIC, I. Position-Based Routing in Ad Hoc Networks. *IEEE Communications Magazine 40*, 7 (2002), 128–134. (Cited on page 12.)

[173] TAKAGI, H., AND KLEINROCK, L. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Transactions on Communications 32*, 3 (1984), 246–257. (Cited on page 3.)

[174] TANG, K., AND GERLA, M. Congestion Control Multicast in Wireless Ad Hoc Networks. *Elsevier Computer Communications 26*, 3 (February 2003), 278–288. (Cited on page 107.)

[175] TANG, K., OBRACZKA, K., LEE, S.-J., AND GERLA, M. A Reliable, Congestion-Controlled Multicast Transport Protocol in Multimedia Multi-hop Networks. In *Proceedings of the Fifth International Symposium on Wireless Personal Multimedia Communications (WPMC 2002)* (Honolulu, HI, USA, October 2002), pp. 252–256. (Cited on page 107.)

[176] TANG, K., OBRACZKA, K., LEE, S.-J., AND GERLA, M. Congestion Controlled Adaptive Lightweight Multicast in Wireless Mobile Ad Hoc Networks. In *Proceedings of the Seventh IEEE Symposium on Computers and Communications (ISCC 2002)* (Taormina - Giardini Naxos, Italy, July 2002), pp. 967–972. (Cited on page 107.)

[177] TANG, K., OBRACZKA, K., LEE, S.-J., AND GERLA, M. Reliable Adaptive Lightweight Multicast Protocol. In *Proceedings of the IEEE International Conference on Communications (ICC 2003)* (Anchorage, AK, USA, May 2003), pp. 1054–1058. (Cited on page 107.)

[178] TOBAGI, F. A., AND KLEINROCK, L. Packet Switching in Radio Channels: Part II–The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution. *IEEE Transactions on Communications 23*, 12 (December 1975), 1417–1433. (Cited on page 3.)

[179] TOH, C.-K. Associativity-Based Routing for Ad Hoc Mobile Networks. *Wireless Personal Communications 4*, 2 (March 1997), 103–139. (Cited on page 30.)

[180] TOH, C.-K., GUICHAL, G., AND BUNCHUA, S. ABAM: On-Demand Associativity-Based Multicast Routing for Ad Hoc Mobile Networks. In *Proceedings of the IEEE 52nd Vehicular Technology Conference (VTC 2000 Fall)* (Boston, MA, USA, September 2000), pp. 987–993. (Cited on page 30.)

[181] TOUSSAINT, G. The relative neighborhood graph of a finite planar set. *Pattern Recognition 12*, 4 (1980), 261–268. (Cited on page 40.)

[182] TRANSIER, M., FÜSSLER, H., BUTTER, T., AND EFFELSBERG, W. Implementing Scalable Position-Based Multicast for the Linux Kernel.

In *Proceedings of the 2nd German Workshop on Mobile Ad-Hoc Networking (WMAN 2004)* (Ulm, Germany, September 2004), pp. 105–110. (Cited on pages 133 and 134.)

[183] TRANSIER, M., FÜSSLER, H., BUTTER, T., AND EFFELSBERG, W. Contention-Based Multicast Forwarding. In *Proceedings of the 7th ACM international Symposium on Mobile and Ad Hoc Networking & Computing, Poster Session (MobiHoc 2006)* (Florence, Italy, May 2006). (Cited on page 85.)

[184] TRANSIER, M., FÜSSLER, H., WIDMER, J., MAUVE, M., AND EFFELSBERG, W. A Hierarchical Approach to Position-Based Multicast for Mobile Ad-hoc Networks. Tech. Rep. TR-04-002, Department for Mathematics and Computer Science, University of Mannheim, 2004. (Cited on page 55.)

[185] TRANSIER, M., FÜSSLER, H., WIDMER, J., MAUVE, M., AND EFFELSBERG, W. Scalable Position-Based Multicast for Mobile Ad-hoc Networks. In *Proceedings of the First International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWIM 2004)* (San José, CA, USA, October 2004). (Cited on page 55.)

[186] TRANSIER, M., FÜSSLER, H., WIDMER, J., MAUVE, M., AND EFFELSBERG, W. A Hierarchical Approach to Position-Based Multicast for Mobile Ad-hoc Networks. *Wireless Networks – The Journal of Mobile Communication, Computation and Information 13*, 4 (August 2007), 447–460. (Cited on page 55.)

[187] TSCHUDIN, C., LUNDGREN, H., AND NORDSTRÖM, E. Embedding MANETs in the Real World. In *Proceedings of the IFIP-TC6 8th International Conference on Personal Wireless Communications (PWC 2003)* (Venice, Italy, September 2003), pp. 578–589. (Cited on page 134.)

[188] WAITZMAN, D., PARTRIDGE, C., AND DEERING, S. Distance Vector Multicast Routing Protocol. RFC 1075, `http://www.ietf.org/rfc/rfc1075.txt`, 1988. Status: EXPERIMENTAL. (Cited on page 6.)

[189] WALL, D. W. *Mechanisms for Broadcast and Selective Broadcast*. PhD thesis, Stanford University, November 1980. (Cited on page 5.)

[190] WILLIAMS, B., AND CAMP, T. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of the Third ACM*

international Symposium on Mobile and Ad Hoc Networking & Computing (MobiHoc 2002) (Lausanne, Switzerland, June 2002), pp. 194–205. (Cited on page 19.)

[191] WISCHHOF, L., EBNER, A., ROHLING, H., LOTT, M., AND HALFMANN, R. Self-Organizing Traffic Information System (SOTIS). In *Inter-Vehicle-Communications Based on Ad Hoc Networking Principles—The FleetNet Project*, W. Franz, H. Hartenstein, and M. Mauve, Eds. Universitätsverlag Karlsruhe, Karlsruhe, Germany, November 2005, pp. 233–258. (Cited on page 34.)

[192] WITT, M., AND TURAU, V. BGR: Blind Geographic Routing for Sensor Networks. In *Proceedings of the Third IEEE International Workshop on Intelligent Solutions in Embedded Systems (WISES 2005)* (Hamburg, Germany, May 2005), pp. 51–61. (Cited on page 87.)

[193] WONG, J. L., VELTRI, G., AND POTKONJAK, M. Energy-Efficient Data Multicast in Multi-hop Wireless Networks. In *Proceedings of the IEEE Workshop on Integrated Management of Power Aware Computation and Communication (IMPACCT 2002)* (New York City, NY, USA, May 2002), pp. 69–85. (Cited on page 35.)

[194] WU, C. W., AND TAY, Y. C. AMRIS: A Multicast Protocol for Ad Hoc Wireless Networks. In *Proc. of IEEE Military Communications Conference (MILCOM)* (Atlantic City, NJ, USA, November 1999), pp. 25–29. (Cited on pages 24 and 30.)

[195] WU, C. W., TAY, Y. C., AND TOH, C.-K. Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS)—Functional Specification. Internet Draft, draft-ietf-manet-amris-spec-00.txt, expired May 1999, November 1998. (Cited on page 4.)

[196] XIE, J., TALPADE, R. R., MCAULEY, A., AND LIU, M. AMRoute: Ad Hoc Multicast Routing Protocol. *Mobile Networks and Applications 7*, 6 (December 2002), 429–439. (Cited on pages 23 and 30.)

[197] YOON, J., LIU, M., AND NOBLE, B. Random Waypoint Considered Harmful. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)* (San Francisco, CA, USA, March 2003). (Cited on pages 45, 77 and 98.)

[198] ZHOU, H., AND SINGH, S. Content based multicast (CBM) in ad hoc networks. In *Proceedings of the First ACM international Symposium on*

Mobile and Ad Hoc Networking & Computing (MobiHoc 2000) (Boston, MA, USA, August 2000), pp. 51–60. (Cited on page 30.)

[199] ZORZI, M., AND RAO, R. R. Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance. *IEEE Transactions on Mobile Computing 2*, 4 (October 2003), 349–365. (Cited on page 87.)

[200] ZORZI, M., AND RAO, R. R. Geographic Random Forwarding (GeRaF) for ad hoc and sensor networks: multihop performance. *IEEE Transactions on Mobile Computing 2*, 4 (October 2003), 337–348. (Cited on page 87.)